Bjarke Aalto, 2017
Aalto University, School of Arts, Design and Architecture
Department of Film, Television and Scenography
Master's Programme in Design for Theater, Film and Television
Major in Production Design
Supervisor: Tanja Bastamow

Open Source and Free software in film and media production.

Aalto University
School of Arts, Design
and Architecture

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Master of Arts thesis abstract

Author Bjarke Aalto

Title of Thesis Open Source and Free Software in Film and Media Production

Department Department of Film, Television and Scenography

Degree programme Master's Programme in Design for Theater, Film and Television, Major in Production Design

Year 2017 **Number of pages** 80 **Language** English

Abstract

The Open Source and Free philosophy encourages an environment of freely sharing ideas, tools and content. Because of this, a user can freely share, customize and build upon existing solutions. Using this philosophy of sharing tools, ideas and content in a film and media production environment, has been an interest of mine for a while now. I am interested in exploring an alternative to the industry standards in the field today and can smaller budget films and freelancers benefit from this?

Software can be put into two different categories, proprietary software and Open Source or Free (as in free speech not free stuff) software. Proprietary software refers to software that is closed source. This means that the software is exclusive property of its developers or publishers and cannot be copied or distributed without complying with their licensing agreements. Almost all commercial software is proprietary. Free and Open Source software are open to modification and redistribution. Because of the open development model of Open Source and Free software, I ask the question: "Can Open Source and Free software be used for professional film production, media production and education?"

The research for this thesis is a combination of practical and qualitative research. First, I study and analyse the philosophy of Open Source and Free Software, with a focus on the implementation of it to film and media production. Secondly, I include real world examples of implementing this software to professional film and media production. Thirdly, I contemplate the ethical questions and possible issues of using non-copyrighted tools to create copyrighted material. Finally, I have included two of my own projects that use Open Source and Free software. These are the Open Creative Suite and the Portable virtual studio system. The Open Creative Suite is my attempt to compile and unify a group of programs that can be used in a pre-production environment. To make the installation and use of them easier to my peers and colleagues. With the Portable virtual studio I attempt to build a portable system to film live footage with a virtual environment. This project is made to test the Open Source and Free software style of development.

The research brought up some interesting points. First, Open Source and Free software has finally reached the quality it needs to be used in professional production. Secondly, it explores the ethical dialogue of the responsibility we as creators have of making our techniques and tools accessible to others. Finland is a small country and to efficiently use the resources we have, Open Source and Free software provides a partial solution.

By the help of this research I hope to shed some light on this subject and demonstrate to others an alternative way of doing creative work.

Aalto-yliopisto
Taiteiden ja suunnittelun korkeakoulu

Aalto Yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Taiteen maisterin opinnäytteen tiivistelmä

Tekijä Bjarke Aalto

Työn nimi Open Source and Free Software in Film and Media Production

Laitos Elokuvataiteen ja Lavastustaiteen laitos

Koulutusohjelma Lavastustaiteen maisteriohjelma, Elokuva- ja TV- lavastuksen pääaine

Vuosi 2017 **Sivumäärä** 80 **Kieli** Englanti

Tiivistemä
Avoimien ja Vapaiden ohjelmistojen filosofia kannustaa ideoiden, työkalujen ja sisällön vapaaseen jakamiseen. Tämän takia ohjelmistojen käyttäjät voivat vapaasti jakaa, muuttaa ja rakentaa olemassa olevien ratkaisujen pälle. Tämän jakamisen filosofian käyttäminen elokuva ja media tuotannon yhteydessä, on ollut kiinnostuksen kohde minulle jo jonkin aikaa. Olen myös kinnostunut tutkimaan vaihtoehtoa elokuva ja mediateollisuuden käyttämiin nykyisiin ratkaisuihin. Voivatko esimerkiksi pienen budjetin elokuvat ja freelancerit hyötyä tästä?

Ohjelmistot voidaan jakaa kahteen eri kategoriaan, suljettuihin ja avoimiin tai vapaisiin. Suljetut ohjemistot viittaavat ohjelmistoihin jotka eivät jaa lähdekoodia. Tämä tarkoittaa että ohjelmisto on yksinomaan sen kehittäjien omaisuutta ja sitä ei saa kopioida, jakaa tai käyttää suostumatta heidän lisenssiin. Melkein kaikki kaupallinen ohjelmisto on suljettua. Avoimien ja Vapaiden ohjelmien vapaamman kehitysmallin takia, esitän kysymyksen: "Soveltuvatko Avoimet ja Vapaat ohjelmat Elokuvan ja median ammatilliseen tuotantoon?"

Tutkimus rakentuu käytännön ja laadullisen tutkimuksen yhdistelmästä. Ensiksi, tutkin ja analysoin Avoimien ja Vapaiden ohjelmistojen filosofiaa, pääteemana miten se soveltuu elokuvan ja median tuotantoympäristöön. Toiseksi, käyn läpi esimerkkejä ammattimaailman ratkaisuista käyttää tämän kaltaisia ohjelmistoja. Kolmanneksi, pohdin eettisiä kysymyksiä ja mahdollisia ongelmia tekijänoikeusvapaan työkalun käytöstä luomaan tekijänoikeutettua materiaalia. Lopuksi, esittelen kaksi omaa projektiani jotka käyttävät Avoimia ja Vapaita ohjelmia. Nämä ovat Avoin Luova Ohjelmistokokoelma (Open Creative Suite) ja Kannettava virtuaalistudio (Portable virtual studio system). Avoin Luova Ohjelmistokokoelma on yritykseni koota ja yhdistää ryhmä ohjelmia jota voidaan käyttää tuotannon ennakkosuunnitteluun. Ajatus tässä on tehdä näiden ohjelmien asennus ja käyttöönotto helpommaksi kolleegoilleni. Kannettava virtuaalistudio on yritykseni rakentaa kannettava järjestelmä jolla voidaan kuvata reaaliaikaista kuvaa virtuaalisessa ympäristössä. Projektin toteutin kokeena, kokeillakseni avoimien ja vapaiden ohjelmistojen kehitysmallia.

Tutkimus tuo esille pari kiinnostavaa huomiota. Ensiksi, Vapaat ja Avoimet ohjelmistot ovat viimeinkin saavuttaneet laadut ne vaativat, jotta niitä voi käyttää elokuvien ja median ammattituotannossa. Toiseksi, se tuo esille eettisen dialogin meidän vastuusta tekijöinä, tehdä tekniikoistamme ja työkaluistamme saatavia muille. Suomi on pieni maa ja pienet resurssit pitää käyttää tehokkaasti, avoimet ja vapaat ohjelmistot tarjoavat osaratkaisun tälle.

Tällä tutkimuksella haluan valaista tätä aihetta ja esitellä kolleegoilleni vaihtoehtoisen tavan tehdä luovaa työtä.

Table of Contents

1 Preface

My initial interest for this thesis was sparked in 2012 when I replaced the Microsoft Windows operating system[1] in my computer to a Gnu/Linux[2] based one called Ubuntu[3]. The reason I wanted to change my operating system came from a need to have control over my computer. This need came from a realization that I was spending more time troubleshooting my computer, than actually doing work on it. Gnu/Linux operating systems are usually more customizable and offer more control to the root user.

My design process is usually very computer centric. Usually, after I do some initial drawings on paper, I bring them in to a computer program, for coloring and enhancement. If a film uses real locations, I usually edit pictures of those locations in an image manipulation program to show how they will be modified, to suit the film. For studio sets I often use a 3D modeling application to design the look and a CAD program for making the build plans. Because of this design method, I really need a computer to do my work. In 2012, I had a laptop from 2008 running Windows Vista[4], that came with the computer. My computer was starting to get very slow and I was having trouble doing my work on it. This was mostly due to the operating system taking up a lot of resources, but because I was a student, I could not afford to upgrade to Windows 7[5] or buy a Macbook, which seemed to be the favored computer in my field. Having tried a GNU/Linux operating system before, I came to the conclusion that it was the best choice for me. It was light on resources and most of the distributions were offered for free by the developers. This gave new life to my laptop and I was able to use it to the end of its hardware life cycle, not just the software life cycle.

Making this change also involved that I had to change the software I was using. A lot of the proprietary programs I had used before were not available on a GNU/Linux operating system. Eventually, I started to realize the potential of using Open Source and Free[1] software and transitioned my whole work flow to using it. The Open Source and Free philosophy encourages an environment of freely sharing ideas, tools and content. Because of this, a user can freely share, customize and build upon existing solutions. In the software industry this has led to many developers creating their own versions or building upon existing programs.

Using this philosophy of sharing tools, ideas and content in a film and media production environment, has been an interest of mine for a while now. I have found some articles and projects that have explored this in the past, but many of them are more aimed toward non-professional work. I was also interested in exploring an alternative to the industry standards in the field today and can smaller budget films and freelancers benefit from this? Finally, I wanted to test if Open Source and Free software could aid when doing research and development of new techniques for film and media production. To share my own research and exploration of these topics, I decided to write my Master’s thesis on them.

Computer software is a very important aspect of the world we live in today. Almost every convenience we have in our everyday life is due to a computer running some kind of

1 Free as in free speech, not free stuff.

software in the background. This software we use can be put into two different categories, proprietary software and Open Source or Free (as in free speech not free stuff) software.

Proprietary software refers to software that is closed source. This means that the software is exclusive property of its developers or publishers and cannot be copied or distributed without complying with their licensing agreements. Almost all commercial software is proprietary. For example, the Microsoft Windows operating system and Adobe Photoshop[6] are both proprietary software. Due to the nature of the software license, used in proprietary software, the end user is not permitted to make any changes or customization's to the software. The user has to trust that the software provided by a company does what it states and it does not contain any security risks.

In contrast to this Free and Open Source software does not contain such limitations. In order for a computer program to call itself “Free”, according to the Free Software Foundation, it needs to fill these four requirements: Freedom to use the software for any purpose, Freedom to change the software to suit your needs, Freedom to share the software with your friends and neighbors, and Freedom to share the changes you make. Software with these Freedoms are usually published under the GNU general public license, or in short GNU GPL [7]. Other such licenses are the Apache license, Mozilla Public license and MIT license [8]. Because Free and Open Source software are open to modification and redistribution, they also have a bigger base of developers than proprietary software. Due to anyone with the skill and knowledge being able to contribute fixes and improvements to the code.

Because of the open development model of Open Source and Free software, I ask the question:

“Can Open Source and Free software be used for professional film production, media production and education?”

1.1 The research

The research for my thesis will be a combination of practical and qualitative research. I will firstly study and analyze the philosophy of Open Source and Free software, with a focus on how to implement this in a film and media production environment. To this there are two approaches. First there is the Free Software Foundation, which focuses strongly on the ethical side. Thus, making Free software a political and human rights issue[9]. Secondly, there is the Open Source Initiative approach, that argues for the use of Open Source software from a very practical sense[10].

Secondly I will include real world examples, of the use of Open Source and Free software in film and media production, IE. case studies. These will include examples such as; Jupiter Broadcasting, a pod-casting network which strives to use Open Source and Free software in the production of their audio and video production[11]; Open movies, for example the short films released by the Blender Foundation, that not only use Open Source and Free software, they also release all the models and footage used in the making of the film. Thus implementing the Open Source philosophy to film-making[12]; Hollywood,

This example is about the use of Open Source and Free software in professional production companies. Such as Industrial Light and Magic, who released an Open Source high dynamic range image file format named openEXR. This has been used in films like *Harry Potter (2001-2011), Men in Black II (2002), Gangs of New York (2002) and Signs (2002)* [13].

The third part of my research will be compiled of two of my own projects, The Open Creative Suite and the Portable virtual studio. The first project is a creative software suite, which includes software that I have transitioned to using in my work-flow, instead of the proprietary software I used to use. In this project I make the attempt to take all these different programs and unify their look and feel. The reasons for this is to demonstrate an alternative work-flow for my peers and colleagues. To make the transition easier for those who want to try Open Source and Free software as well as providing a precompiled package.

The second project, the Portable virtual studio, is a demonstration of how we can benefit from the use of Open Source and Free software. For this demonstration, I make the attempt to build a portable version of the virtual studio system in Aalto University Studios[2]. To achieve the portability aspect I have chosen to use Internal measurement unit sensors[3] coupled with an Arduino[4] microcontroller instead of the infrared cameras[5] used in the existing studio. The point of this project is to base a lot of the software required on existing code, provided by the Open Source community and build upon it to make an application for filming live footage in a virtual environment.

By the help of this research I hope to shed some light on this subject and demonstrate to others an alternative way of doing creative work.

2 This refers to the studio in Aalto University capable of tracking motion with infrared cameras. By using software like Brainstorm's eStudio[14] students are capable of filming live footage within a virtual environment. The studio is capable of techniques like motion capture and live compositing. Studios like this are often used in news and election coverage. The Finnish Media network YLE also has a similar studio setup in the city of Tampere. This studio is not to be confused with the Virtual Studio space in Aalto Universities Otaniemi campus.

3 In short IMU. An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. IMU's are typically used to maneuver aircraft, including unmanned aerial vehicles (UAV), among many others, and spacecraft, including satellites and landers.

4 Arduino is an Open Source, computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world.

5 The motion capture (or infrared) camera is designed for a very specific purpose: to capture a narrow spectrum of high-intensity light at very fast speeds. This light is emitted from the strobe, reflected back by the marker, and focused by the lens through the filter onto the image sensor.[15]

1.2 The arguments

According to the Open Source initiative there are several reasons why Open Source and Free software is superior to proprietary. The first and one of the most popular arguments is security. The argument is based on the source code being available for public audit. This means that the end users, experts, and the Open Source community at large can verify that the software does exactly what it claims to do and that there are no back doors[16]. Another argument toward this is that Open Source and Free software has a bigger community of developers, which leads to faster bug-fixes and new features[17].

Another argument for Open Source and Free software has to do with the freedom to modify software. Because of this freedom a user may use the existing code of a program and customize it to suit their needs. With this freedom they can also use existing code and create a completely new program, basing it on a working solution. This leads to an increase in productivity, for the user is not spending time "reinventing the wheel". This technique of development is very popular in GNU/Linux distributions[6]. For example, the operating system Ubuntu is originally based on code from the operating system Debian[19][20].

Other arguments that are often used to advocate Open Source and Free software include: due to the freedom to share the software, many of the programs are Free in the sense of "Free stuff" as in "Free speech"; many of the programs are platform agnostic; a lot of the programs are very resource efficient, which leads to less waste, as hardware needs to be upgraded less frequently.

I have included education in my question. The reason for this has to do with the software that is used in the education in media and film schools. For example, the majority of the workstations in Aalto University are running Microsoft Windows. The argument regarding the use of Open Source and Free software in education is, unlike the other arguments, an ethical one. According to The Free software Foundation, educational institutions of all levels should use and teach Free software because it is the only software that allows them to accomplish their essential missions: to disseminate human knowledge and to prepare students to be good members of their community. The source code and the methods of Free Software are part of human knowledge.[21] Dr. Richard M. Stallman writes, in an essay on the subject, that when educational institutes teach students non-Free software, they implant a form of dependence on them . After all, the companies that do offer free student versions, only do so to recruit new users to their product.

6 A Linux distribution (or distro) is an operating system made from a software collection, which is based upon the Linux kernel[18] and often a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices and personal computers to powerful supercomputers. A typical Linux distribution comprises a Linux kernel, GNU tools and libraries (should then be called GNU/Linux distribution), additional software, documentation, a window system, a window manager, and a desktop environment. Most of the included software is Free and Open Source software made available both as compiled binaries and in source code form, allowing modifications to the original software. Usually, Linux distributions optionally include some proprietary software that may not be available in source code form, such as binary blobs required for some device drivers.

1.3 Fundamentals

To understand the themes and concepts in this thesis you don't need to be a software developer, but you need to know some fundamentals of what programming is and how programs work. In the following segment I will go through in short: what is a computer program; what source code is and; the difference between proprietary, Open Source software and Free software.

A computer program is a set of instructions that performs a specific task when executed on a computer. Without computer programs, a computer can not function and typically executes the programs instructions in a *central processing unit*, or in short CPU. To simplify things, a computer can be viewed as a light switch. It has an ON setting and an OFF setting, or 1 and 0. Now by switching a light ON (1) and OFF (0) you can get Morse code. For example, if you switch your light ON and OFF three times, or 010101 In Morse code this means S. This is basically what a computer does. It takes an input of 0 (OFF) and 1 (ON) and provides a certain output. All computer programs are made of a series of combinations of 0 and 1, this is called binary code.

Now, writing a program in binary code is extremely complicated. Thus, software developers have invented something called programming languages. These are instructions written in a way that is understandable to humans. They are then translated into binary code using a compiler[7]. For example, here is a simple “Hello World” program[8] written in a programming language called C:

```
#include <stdio.h>

      int main(void)
      {
        printf("Hello World\n");
        return 0;
      }
```

This might for some still look very complicated, but it is easier to understand than the binary code version. Here is the same program in binary code:

01001000 01100101 01101100 01101100
01101111 00100000 01010111 01101111
01110010 01101100 01100100

These two versions of the same program can also be referred to as the precompiled and compiled. The precompiled version, IE. human understandable, is also known as “source code”.

7 A compiler is a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code.

8 A "Hello, World!" program is a computer program that outputs or displays "Hello, World!" to a user. Being a very simple program in most programming languages, it is often used to illustrate the basic syntax of a programming language for a working program. It is often the very first program people write when they are new to a language.

To summarize, a software developer writes a program in a programming language, this is then translated into binary code, which the computer understands, but is non readable to humans.

The term proprietary software refers to programs that are only released by the developers in compiled form. This means that the users of the software are only provided with the end result of the developers work, but they have no way of verifying how the software works.

Free and Open Source software is released with the source code so that the user can read it as well as study how the program works and is built.

2 Why is this relevant?

As this is a very software and computer oriented thesis, the question may arise how this is relevant to making cinema?

Film-making has experienced a digital revolution since the year 2000. Most of the visual media we consume and produce today are in a digital format. Also, most of the tools we use in the production pipeline are computer programs. For example the preferred tools for video editing are usually: Final Cut Pro[22], Adobe Premiere[23] and Avid[24]. In the same manner, Production designers have an array of digital tools to help them realize their design and vision. These tools usually range from 3D modeling software to illustration software such as Maya 3D[25] and Adobe Illustrator[26] or even Adobe Photoshop. All the software mentioned above is proprietary.

The fundamental question of relevancy, is not really relevant to software use, but the idea of sharing knowledge freely. In small countries such as Finland, there are not enough resources to sustain large production companies. This also means that smaller production companies have less resources and might not be able to, for example, develop their own software. Unlike bigger studios like Pixar or Industrial Light and Magic. This leaves the companies with the option to use a software provided by a vendor like, for example, Autodesk. These programs are usually proprietary software and include all the limitations caused by the license agreement. These limitations are limited customization, limited use case, platform dependency, hardware limitations etc.

Open Source and Free software offers an alternative to this. Because the license agreement makes the software public property, many of the limitations are diminished. This means that the small production companies can collaborate together in the development for specialized tools. Building upon an existing platform. The general idea is that if these small companies collaborate, they could together build tools capable of the same quality as the bigger studios. There are many ways these studios could collaborate, but the main way could be the usage and funding of Open Source and Free software.

For example, Tangent animation, a Canadian production company, released an animated feature film *Ozzy, 2016* made entirely with Blender[27]. *Blender is a professional Free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender is a professional Free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games.* In a presentation, held at Blender Conference 2016. Two of the founding members of the company (Jeff Bell and Ken Zorniak) explained that they were able to reroute all the funds spent earlier to buy Autodesk's Maya 3D to developers and artists as well as contribute back all the development related to Blender. They also sponsored two seats in the Blender foundation. Bell also explained that because they spent the money on developers and not buying Maya, the money was tax credible in Canada. Thus, resulting in a budget increase[28]. A similar Tax incentive program was introduced in Finland in January 2017. The tax incentive states that an audiovisual production can apply to refund a part of their expenses, that have been spent on wages and service purchases in Finland[29]. This means that a Finnish production company can instead of

buying a license to a certain proprietary software, hire a Finnish developer to build upon a Free or Open Source software. Since the budget is now going to the developers wages and not a license purchase, the company can apply to get a part of the wages refunded under the tax incentive.

Tangent animation was as a single company able to improve the Cycles render engine in Blender by a great deal. This invokes a question of the possibilities, if we collaborated more and had many companies working for a single goal.

Open Source as a philosophy and fundamental idea is something that could be explored outside the software sphere. If the idea would be explored in the constrains of cinema and film production, work-flows and even raw content could be released freely to the public after a project finishes. This could also include the free release of the actual film, but since that would affect the income of the film. Thus, consequently the financial income of the creators, I will leave it out for this argument. The release of the raw material on the other hand would possibly not affect the income of the end product. Since the actual assembly of a copyrighted film from the raw material, would result in a product too similar to the end product. Thus, leading to legal action for committing piracy. On the other hand, the creative and educational impact of releasing the techniques and raw material could lead to an improvement of the craftsmanship and increase in the creative content. For example, if Disney would release all the raw material of *Star Wars (1977)*, considering the popularity of the franchise, there would be an explosion of fan made content based on that material. This could lead to someone creating a meaningful piece of art, because people are not constrained by copyright law. To protect from misuse, the raw material could be released under a share alike license[9]. To put it another way, this scenario could be explored through other art forms, like pop art. For example, if there had been restrictions on using Campbell's soup cans, Andy Warhol might not have created his now iconic paintings. The release of the techniques used could also help further the knowledge and craftsmanship in the industry and lead to new techniques, as others could build upon the existing ones.

Most importantly, are we as creators ready to move from a self-serving working ethic to a community serving one? This is something we can explore through the community of Open Source and Free software.

Another part of this is education. This matters because the educational institutes produce a big part of the next generations of creators and artists. We all have a responsibility to teach them working methods and techniques that do not restrict, but empower them. For me, the basic principle of education, is to teach younger generations, knowledge, craftsmanship, tools and techniques. That older generations have developed. These things should not come with restrictions or heavy price tags, but be free and open to build upon. Think of it like this: when an educational institute teaches a specific tool, it has the potential to become the industry standard, because everyone coming from the institute knows how to use it. That puts a big burden, or responsibility, on the institute to choose a tool that is available to all the students. Choosing a tool that is freely available or Open Source provides a solution for this.

9 Share-alike is a copyright licensing term, originally used by the Creative Commons project, to describe works or licenses that require copies or adaptations of the work to be released under the same or similar license as the original. Copyleft licenses are Free content or Free software licenses with a share-alike condition.

2.1 The arguments with software

The fundamental difference of proprietary and Open Source and Free software is that in the former's development process all the source code is only accessible to the insiders of the company developing the software. While the latter option, makes all the code public knowledge. This difference has over the years led to many arguments on both sides, stating that their process leads to better software.

The arguments for both sides are mainly security, stability, user-friendliness and trust in the software.

The security of a software is usually the most important part of a program and developers on both sides spend a lot of time on this. To make a program secure, means that when it is under a malicious attack. For example, from a malicious hacker[10]. It will still continue to function correctly. The ways a malicious hacker attacks a system can take many forms. These ways are often known as Malware[11], Viruses[12], Trojans[13] and Zero day attacks. Zero day attacks are one area were the differences in proprietary and Open Source and Free software are very clear.

Zero day attacks refer to a vulnerability in a program that a malicious hacker can use to gain access to a network or system. These vulnerabilities are often caused by design flaws in the source code.

The arguments for Open Source and Free software is that, because the source code is open to public audit, there are more people checking the code. Thus, more people looking for flaws in it. Coupled to this is the contribution system used by many Open Source and Free programs. This system works in the following way: if a user finds a bug or a flaw in the program, they can submit a bug report to the developers. Another user sees the bug report, finds a fix for it and contributes the code to the developers. This then allows the developers to review and incorporate the fix to the program. This usually leads to faster fixing of flaws and bugs in Open Source and Free software.

This argument works on the prerequisite that there are enough people using and reviewing the software, that posses the right skill-set to understand the source code. Which means, that the more popular the software the more review it receives. Therefore, if a program targets a niche market it might not have enough users to adequately review it.

The same argument can also be reversed to argue for proprietary software. Since the source code is not public knowledge, it is harder for a malicious hacker to find vulnerabilities. The

10 I have chosen to use the term malicious hacker, because the term hacker refers to anyone who programs software. This term is often used incorrectly to describe a person who uses programming to cause harm.

11 Malware is a general term shortened from “malicious software” and is a general term used for a group of programs including, computer viruses, Trojans, Worms, Ransom Ware, Spyware, etc. These programs are usually used together by a malicious hacker to gain access to a user's system and steal data.

12 Viruses refer to computer programs that when executed, replicate themselves by copying their own source code or infect other programs by modifying them. The term computer virus is also sometimes used in place of the term “Malware”.

13 Trojans or Trojan horses, are programs that misrepresent themselves to appear friendly in attempt to persuade a victim to install them. They can for example be hidden in an e-mail attachment.

argument does make sense in theory, but there are ways a hacker can get access to the code. For example by using an Interactive Disassembler[14] to reverse engineer the code.

The arguments on security are further debunked by the fact that Open Source, Free and proprietary software experience zero day hacks every year. Some vulnerabilities are also non- exclusive and affect both proprietary and, Open Source and Free software.

There are other factors that affect the security of a software. A program can be very well-designed but still experience attacks. Microsoft Windows for example experiences a very large part of the attacks done on systems, but it also holds the largest part of the market share in desktop operating systems. Since it is the most popular operating system, it also makes sense to a malicious hacker to target it.

The conclusion of this is that the theoretical arguments could be declared irrelevant, as proprietary, Open Source and Free software experience security attacks and vulnerabilities. Therefore, the theoretical arguments do not show a correlation in practical use.

The second main argument for Open Source and Free software is stability. The argument is that since the source code is public knowledge, the software is protected against drastic changes or disappearance. This can be very relevant in software for industry use.

Using proprietary software as industry standards has some issues involved with it. First of all, when using proprietary software a user is completely subjugated to the software providers ideas and decisions. In other words, if the software provider decides to redesign or discontinue a program, the user is left with either to learn a new program or use an obsolete program. This for example happened when Apple updated their video editing software from Final Cut 7[30] to Final Cut X[31] in 2011. The initial update included a complete redesign of the programs GUI[15], a regression in features and a magnetic time-line in favor of the old track based one. This left a lot of users frustrated and forced to change their work-flow. The production company Bunim/Murray for instance decided to move their whole editing work-flow to Avid[32].

When using Open Source or Free software the risks for the scenario described above are diminished. The reason for this is if a program is changed or discontinued by the original developer, another developer can create an own version. For example, with an old GUI design or features no longer supported by the original. Another developer can also continue to support a program that has been discontinued by the original. This happened to Ubuntu in 2010 when Mark Shuttleworth started the Unity project. Before the Unity project Ubuntu used a desktop environment called Gnome[34] as its default. If Ubuntu had been proprietary software the users would have had to learn a whole new desktop environment, or change to another operating system. Since this is not the case, Ubuntu users who did not like the new desktop were free to install the Gnome desktop themselves. In 2012 a group of developers also released a flavor[16] of Ubuntu called Ubuntu Gnome[35].

14 The Interactive Disassembler (IDA) is a disassembler for computer software which generates assembly language source code from machine-executable (binary) code.

15 Stands for Graphical User Interface.

16 Distributions based on Ubuntu are called flavors. They all share the common Ubuntu repositories for downloading updates but feature a different set of packages installed, like other desktop environments

This is also something a user can directly influence by choosing a software provider that they trust to keep the software working in a fashion that suits their needs. Sometimes changes can also lead to a better more robust program. This happened to Blender when they released the 2.5 version of their software. The release featured a complete remake of the program's user interface. This redesigned user interface was made because the developers wanted to make Blender more user-friendly and simple to learn. Even if Blender is Free software, the same can happen to proprietary software.

User-friendliness is another part were there are big differences in programs. This is of course a very subjective thing and varies from user to user. Considering this, some solutions have proven to be more popular than others. For example, it is very hard to find a graphical user interface or operating system that does not use the traditional window manager[17] layout to browse files.

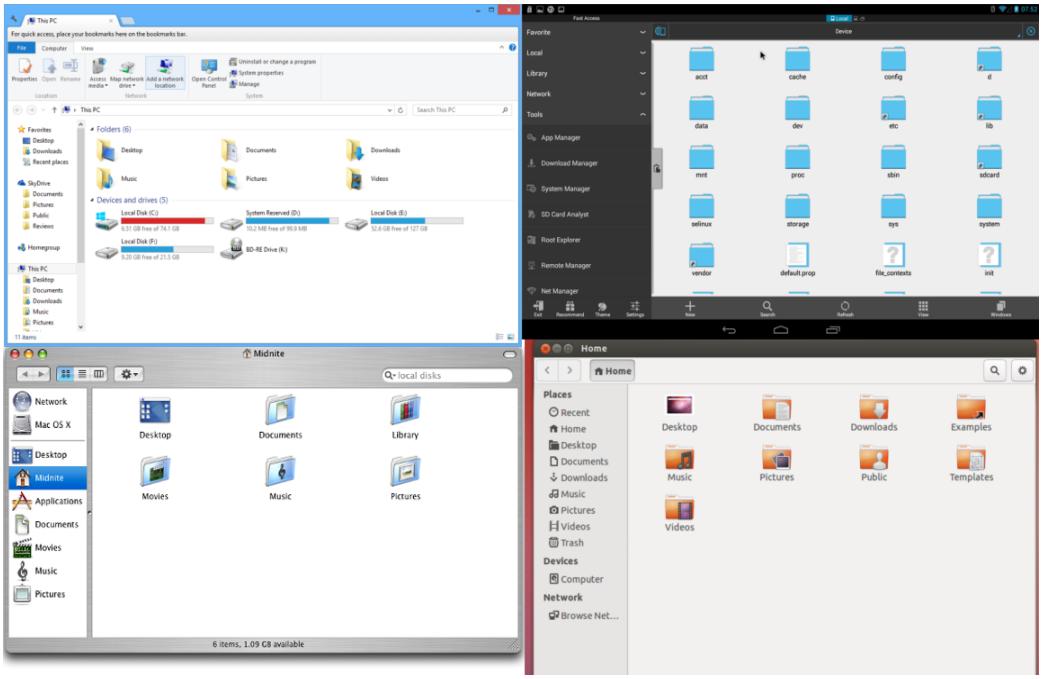


Illustration 1: From the left, Windows, Android[36], Mac OS[37] and Ubuntu

User-friendliness also includes many other parts, not just the actual user experience. Another very important part of user-friendliness is support. Having good support for new users is essential in having a user-friendly program. This is also an area were many Open Source and Free programs fall short. This is mainly because most of the support comes from the community and other users surrounding the software.

There are some projects that offer very good support in the Open Source and Free landscape, but in general a user is often left to solve things by finding the solutions themselves. There are also steps a user can take to minimize the issues they will experience

than Unity[33].

17 A window manager is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface. Most window managers are designed to help provide a desktop environment.

when choosing what software to use. For example instead of installing a GNU/Linux distribution by themselves on a computer that came with a proprietary platform, they can buy a computer with it preinstalled. Many of the vendors that offer these computers also offer the same kind of support and help desk systems as vendors of proprietary systems. Other projects like Red Hat Inc. offer support with their Open Source operating system (Red Hat[38]). This is of course a paid support. Which is why Red Hat is one of few GNU/Linux distributions with a price tag. If a user can't pay for support and wants a free (as in free stuff) distribution, they can always go with something like Ubuntu that has a very good community support system.

Some issues in Open Source and Free software have to do with funding and popularity. The issue is this: the more people that use Open Source and Free software the more vendors and manufacturers will offer support for them, but before there is good enough support and availability, people will not start using the software in mass. Many Open Source and Free programs require a lot more learning and reading from the user. Especially when using more complicated systems like external or hybrid graphics[18].

In contrast, many proprietary programs offer a support or help desk system with the purchase of the license. This makes the variation in the quality of the support much smaller. In general user-friendliness is better with proprietary software, because of larger funds.

Trust is the last main argument for Open Source and Free software. This is something that is actually a verifiable argument, since the source code is public information. Open Source and Free software works on a "trust but verify" system, because of this the developers and software they provide often feel more trustworthy. As a contrast, proprietary software vendors ask a user to trust them without the same basis.

In other words, when using proprietary software the user has no way of knowing what is included in the program they buy from vendors or if it is as safe as the software company states. The only thing the user can do is not use the software or trust that the company, driven by revenue, is honest in its intentions. Some other things can of course help a proprietary software vendor to increase the users trust, but there is always a doubt of their true intentions involved. For example, the FBI- Apple encryption dispute in 2015. In this dispute Apple refused to unlock an IPhone 5C used by a participant in the San Bernardino attack when asked by the FBI[39]. This could be interpreted by users that Apple will protect their users privacy even when pressured by the government. But is it because it would affect their Phone sales, or because they care about the user?

Business corporations, or companies, are mostly driven by a sole purpose of making money. This also means that these business entities are susceptible to "immoral" acts if they lead to more revenue. For example, the privacy statement of Microsoft states that they collect a number of user data with their software. Including: name and contact data,

18 Hybrid-graphics is a concept involving two graphics cards on the same computer. The laptop manufacturers developed new technologies involving two graphic cards in a single computer, with different abilities and power consumption's. Hybrid-graphics is developed to support both high performance and power saving usages.

credentials, demographic data, payment data, usage data, interests and favorites, contact and relationships, location data and content.[40] This data is mostly used to target advertisements to the user. They also state that they will share all of this information with their affiliates, subsidiaries, vendors and when required by law. This creates an issue in privacy as a revenue driven company is basically allowed to record and sell everything the user does on a computer running their software. This user data collection is also done by many other companies in the software business. Google inc. for one, bases the bulk of its revenue on targeted advertisement.

Open Source and Free software is mostly funded by donation based systems. With some exceptions like Red Hat Inc. Many of them are also very transparent with their work-flow. Not only making the source code public, but also letting users listen to their developer meetings and read their emails regarding the development. All this transparency coupled with the funding system makes it very hard for these developers to hide anything malicious in their software and creates an environment of trust.

Another area were proprietary software usually carries the upper hand is specialized tools. There are many areas in the creative and industrial field that require a specific software for a certain job. These tools can sometimes be used internally by a single company. Open Source and Free software are many times more general in their user specifications and can often be used for more than one purpose. All of this, has to do with the target audience size. Since many Open Source and Free programs rely on the community, the size of that community also impacts the potential quality of a certain software. I am not stating that there is not an Open Source or Free alternative to a specific proprietary program, but that the quality of that program might not be up to the same level. This is mainly because of Freedom 3 (the Freedom to share Open Source and Free software). Because of this Freedom, Free and Open Source software rarely carries the same price as the proprietary alternative. If the software targets a specific use case, its audience might not be big enough to fund (by donation) the development process adequately. This means that the developer can't spend the same time developing the software and that can lead to a lacking product. This is why many Open Source and Free advocates underline the fact that even if a software is Free (as in Free stuff) a user should always donate to the project if possible, especially if it is in daily or professional use.

Sometimes, abundance of choice is used as an argument for the use of Open Source and Free software. While this is true on the topic of operating systems, it falls short in other areas. For example, on the proprietary side, Windows and Mac OS are the two most usual choices on desktop operating systems. While on the Open Source and Free side, there are an abundance of systems to choose from. Like Ubuntu, FreeBSD[41], Arch[42], CentOS[43], Debian, Solus[44], etc. Granted, most of these choices are based on the GNU/Linux system but many of them use very different solutions for the same things. On the other hand when a user needs something specialized, like a software for CAD or 3D modeling, the choices are greatly diminished. Especially if a user needs something capable of professional work. Proprietary software vendors on the other hand, usually carry at least two different programs for the same job. For example, FreeCAD[45] is one of the few Open Source and Free programs capable of professional architectural design. On the proprietary side, vendors offer many alternatives for this. Like AutoCad[46], TurboCAD[47], SketchUp pro[48], etc.

Proprietary software does lack in some other aspects. Because there is a dependence on the software, the user is left dependent on the software provider to keep updating and making the software they are using. Another issue is that proprietary software can not be modified. For example, a program may be the industry standard but lack a certain feature the user needs.

Open Source and Free software do not carry these issues. With the availability of the source code they can be modified and maintained by anyone with the knowledge. This also means that the user is in complete control of the program.

In conclusion here is a table with the pros and cons:

	Open Source and Free	Proprietary
Security	+-	+-
Stability	+	-
User Friendliness	+-	+
Support & Help	-	+
Trust	+	-
Specialized tools	-	+
Choise	+-	+-
Customization	+	-

2.2 Community versus monopoly

As I mentioned in an earlier segment, proprietary software is mostly developed by software companies. Companies are in constant competition with each other for customers. In an ideal world this competitive nature would lead to better products, but this is not always the case. In 1999 for example, Microsoft was declared a monopoly by judge Thomas Penfield Jackson. This case was later settled by Microsoft with the Department of justice in 2001[49], but it does bring forth an issue with using software provided by a revenue driven company.

Because companies are in competition for customers, a user can never trust that the software they are using is always going to exist or stay the same. For instance two companies that make software for a certain industry, can be in rivalry for the same customers. Under time, company number 1 may decide to buy company number 2. Thus, acquire a monopoly over the software that the industry uses. Now company number 1 is Free to decide what price it asks for its services to the industry and the users are left with using an overpriced software.

Right now in 2017 there are two companies that dominate the creative software market, Autodesk and Adobe. Autodesk has over the years been acquiring their competition and integrating their technology to their software. For example, in 2008 Autodesk bought Softimage[19] [50] from then owner Avid. In 2014, they announced that the 2015 release will be the final release of the software[51]. Users of Softimage were offered a migration path to Maya or 3DS Max. The announcement to discontinue Softimage was met with dismay

19 A 3D modeling software originally released in 1988 that was in competition with Autodesk's Maya and 3DS Max. It was once considered the entertainment industry standard. For example, the dinosaurs in *Jurassic Park (1993)* were created with this software.

and anger by the VFX community and even resulted in a petition to save the software which 5843 people signed. Autodesk also acquired Maya 3D in 2006, which back then was one of 3DS Max[52], Autodesk's own 3D modeling program, biggest competitors. From 2001 to 2016 the company has made 39 Acquisitions of other software companies.

Adobe is also guilty of similar tactics. In 2005, for example, the company bought its biggest competitor Macromedia and proceeded to discontinue their products like Freehand[53]. It has not received updates for many years. Autodesk and Adobe have also moved to subscription based purchasing. This means that their customers never really own the software, they merely pay to be allowed to use it.

On the Open Source and Free software side this kind of monopoly is virtually impossible. This is because the software is not owned by companies, but are community property. The Blender project for example is developed by the Blender foundation, an independent non-profit public benefit corporation. It is funded by a donation based system and accepts contributions to Blender from the community. Since Blender is released under the GNU GPL version 2 or later, it is also protected from becoming proprietary. Thus, making it impossible to be part of a monopoly. The GNU GPL also states that any modified version must be redistributed as Free software to protect the rights of the software users. This means that if a software company would copy the Blender source code and redistribute it as proprietary software, they would be in violation of the license.

There are also downsides in not having industry standards. For example, GNU/Linux distributions are fairly hard to port software to. This is mainly because many of the distributions have their own package manager[20]. This means that if a software developer wants to make software for the GNU/Linux platform, they also have to package it for all the main distributions. This has seen some change in the form of Snappy Packages[21] [55] but a universal option has yet to be agreed upon.

2.3 The benefits in media education

According to the department of film and television in Aalto university, their aim is to offer an education that gives the students a capacity to work professionally in their field and to exercise influence in it[56]. Other film schools also make similar claims. For example, The New York Film Academy[57]. This is probably the aim of most departments in many universities. The problem with this kind of statement is that when a university teaches primarily proprietary software their aim is not really achieved. Of course there is no obligation for them to teach anything else than the so called “industry standards”. But if they truly wanted to produce filmmakers that are independent and ready to hit the ground running, they should teach the use of Open Source and Free software.

20 A package manager or package management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system consistently.

21 Snappy is a software deployment and package management system originally designed and built by Canonical for the Ubuntu phone operating system[54]. The packages, called 'snaps' and the tool for using them 'snapd', work across a range of Linux distributions and allow therefore distro-agnostic upstream software deployment. The system is designed to work for phone, cloud, Internet of things and desktop computing.

The reason for this is that when a student graduates they might be sufficient in knowledge, but they are lacking in tools. This is because many film schools primarily teach proprietary software. For example, the program curriculum's of several film schools, including The New York Film Academy[58], The Vancouver Film School[59] and The National Film and Television School[60], state that they teach Autodesk's or Adobe's (or both companies) software. The software both these companies offer can be very expensive for someone just out of school. For example, the cost of an Autodesk Maya license is 1.936Eur a year. Additionally, Autodesk and Adobe both use subscription based plans. This means that a user never truly owns the program. The Open Source and Free software alternatives for many programs the companies offer could be Blender, GIMP[61] and FreeCAD.

The department of scenography in Aalto university has recently made some changes to this. They are for example teaching Blender for 3D modeling and strive to inform students on Open Source and Free software alternatives when possible.

3 Open Source or Free software?

When using the terms “Open Source” and “Free” software it is important to note that these do not mean the same thing. Free software is a term used by the Free software Foundation, led by Richard M. Stallman. This movement started its campaign for software users Freedom in 1983.

The term Open Source on the other hand, was created by the Open Source initiative in 1998. Both of these organizations share many of the same goals, but they have a very different approach.

3.1 Free software Foundation

The Free software Foundation (FSF) has a very political approach and makes Freedom of software a human rights issue. Their philosophy states that it is unethical to use proprietary software because it gives its developers power over its users. In order to call a program “Free” it must include four essential Freedoms:

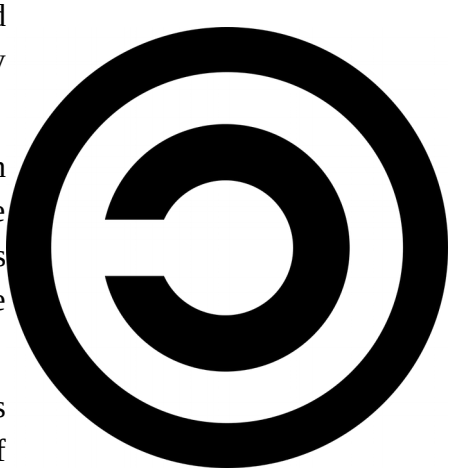
(0) The Freedom to run the program as you wish, for whatever purpose.

(1) The Freedom to study the program's “source code” and change it.

(2) The Freedom to make and distribute exact copies when you wish.

(3) The Freedom to make and distribute copies of your modified versions, when you wish.

According to Richard M. Stallman and the FSF it is important to not use proprietary software, as it is often Malware or spy-ware[62]. This means that the software is often designed to spy on the users. According to him this is a violation of privacy. Thus, steps on the human rights of the user. One example of this could be Microsoft Windows 10, which includes many programs for data collection.



The Free software Foundation and Richard M. Stallman are responsible for many of the key contributions to the Free and Open Source community. These contributions include the GNU General Public License(GPL), the GNU Operating System[63] and Copyleft.

The GNU GPL is a Free software license that guaranties all the Freedoms listed above to the user. It is also one of the most popular licenses used in Free and Open Source software today. This license was

originally written in 1989 by Richard M. Stallman to use with the programs released for the GNU operating system. He based it on an earlier license written for a program he developed, called GNU Emacs[64]. He also demonstrated a way to use the existing Copyright law to create a system of communal ownership. This license has over the years been developed further and Stallman uses the same version system that many programs use. The newest version is GPL v3.

To ensure that the GPL license also protects the software Stallman used a legal technique nicknamed “Copyleft”. This technique helps protect Free software from being exploited, by requiring that all modified and extended versions of a Free program must be released as Free software as well[65]. By doing this a developer can ensure that the software will always remain Free and not be released as proprietary software by anybody else.

Lastly the Free software Foundation with Stallman are the developers of the GNU project, a Free Unix like operating system. The GNU project was initially started by Richard M. Stallman in 1984. This operating system is the foundation for most Linux distributions[22] used today. When it is used in unison with the Linux kernel, developed by Linus Torvalds.

The philosophy of the Free software Foundation is very strict and many programs that are viewed as Open Source are not recognized as Free software by the foundation. For example, the very popular GNU/Linux distribution Ubuntu, is not Free software. One of the biggest reasons for this, is that Ubuntu also offers proprietary software in their software repositories. Such as graphics drivers. Thus, it fails to follow the Free system distribution guidelines (in short GNU FSDG). The GNU FSDG specifically states that: “Some applications and drivers require firmware to function, and sometimes that firmware is distributed only in object code form, under a non-Free license. We call these firmware programs “blobs.” On most GNU/Linux systems, you'll typically find these accompanying some drivers in the kernel Linux. Such firmware should be removed from a Free system distribution.”[66]

Richard M. Stallman also preaches a philosophy of “Freedom over convenience”. The idea of this, is that we as users should value our Freedom over the convenience of simple, practical and powerful software. Smart phones are a good example of this. These devices have become a very large part of many peoples lives. By including networking, phone capabilities and many other features in a hand held device. These “pocket computers” are almost exclusively proprietary software and according to the Free software Foundation “Phones represent one of the most locked-down, proprietary, and generally non-Free technologies in wide distribution.”[67]

22 The correct name for these distributions is GNU/Linux and it refers to operating systems that use the GNU operating system with the Linux kernel. When Linus freed his kernel in 1992 the GNU project used it in their operating system, because their own kernel was not finished. This led to the GNU/Linux operating system which often is shortened to just Linux in conversation, much to the dismay of Richard M. Stallman.

Even if smart phones are mostly proprietary software, they also make many things in our lives simpler and are at times very practical to have. Devices and software like this make actually implementing this philosophy of Freedom over convenience very complicated, because sometimes it would actually cause a regression in technology. It is also something that can become very complicated to users who are not technically inclined. Sometimes it can require knowledge of software engineering. The example of smart phones also brings up another issue, proprietary hardware. To answer this the Free software Foundation has created the *Respect your Freedom* hardware product certification. This program encourages the creation and sale of hardware that respect a user's Freedom and privacy as well as ensures that a user has control over their device[68].

Although Richard M. Stallman and the Free software Foundation have contributed many things to the Free and Open Source community and actively defend the Freedom of computer users, they do carry a stigma. Many members of the Free and Open Source community see their views as too extreme, negative and anti social. The Free software Foundation has also criticized big businesses and companies very openly. This makes the promotion of Free software to business use very hard. Many people have also expressed the opinion that the activist and dogmatic nature can actually hurt the adoption of Free software.

This led to the founding of the Open Source Initiative, as some members of the community wanted to create a business friendly way of promoting Free and Open Source software.

3.2 The Open Source Initiative

The Open Source initiative is an organization with a focus on the pragmatic side of using Open Source and Free software. It was jointly founded by Eric Raymond and Bruce Perens in 1998. At a strategy session in 1998 the organization coined the term “Open Source”. While the Open Source initiative share many of the same goals of the Free software Foundation, it actively avoids the political and activism aspects that are the central arguments for the Free software Foundation. Instead of campaigning against proprietary software the Open Source initiative strives to increase the use of Open Source and Free software by arguing the practical advantages of using it. These advantages include that using Open Source and Free software offer businesses more innovative, secure, productive and strategic tools for their companies. It actively campaigns for commercial use of Open Source and Free software as well as tries to create a more business friendly environment than the Free software Foundation.

Even if the Open Source does not share the ideology of Free software, they do require the same rules from a software to call itself Open Source. These rules are listed in the Open Source definition, that was inspired by the Debian Free software guidelines. These rules include:

- Free redistribution
- Access to the source code
- License to modify and create derived works
- Protect the integrity of the original authors source code
- Its license must not discriminate against any persons or groups
- Its license must not discriminate against any field of endeavor
- The rights attached to the program must apply to all to whom the program is redistributed
- Its license must not be specific to a product
- Its license must not restrict other software
- Its license must be technology-neutral

3.3 Free software in media

The argument of software being a human right has become relevant as the human race keeps moving toward a more computerized world. For example one could argue that governments should use exclusively Free software. This would for example build upon the public's trust, as the software is not developed by a corporate entity.

For media production this issue of Freedom in software could be redeemed as irrelevant. Why would a media production company not just use the best software for the task, be it proprietary or Free? One argument could be that it depends on the media they are producing. If it is fictional film or TV it is completely irrelevant if a company uses Free software or proprietary software, save for good karma. But if the media a company is producing is news or documentaries it is a very relevant issue. The most important argument for this is the integrity of news providers. To be able to provide corruption Free news to an audience a news production company should be completely free of corporate and government influence. This coupled with a community or donation based funding and completely transparent policy, would ensure an audience that the news they are receiving is trustworthy. Most of the news now days is funded either by government or by selling advertisement space. This creates an element of distrust between the audience and the news provider. A Reuters study in 2016 found that trust in the news media is lowest in countries

that are considered to have a strong business and commercial influence over the news. Also, the countries that scored high in political influence, scored low in trust. For example in Hungary where the media is highly politicized; only 14% of Hungarian respondents agreed that the media were independent from undue political or government influence most of the time. Also, only 31% of Hungarian respondents agreed that you can trust most of the news most of the time[69]. In Finland the government funded news YLE, is under risk of censorship when airing news criticizing the government. This for example happened in 2016 when the Finnish media production company YLE was denied to air criticizing news stories about the Finnish prime minister Juha Sipilä[70].

The same argument is relevant when making documentaries. To ensure that a documentary maker is completely free from bias, they should exclusively use Free software and use non corporate or government funding. To build upon the trustworthiness a documentary film-maker could also release all the uncut footage to ensure the public of no alteration of statements in the editing.

3.4 Open Source in media and film production

In this field, The Open Source initiatives pragmatic philosophy is very relevant. It is very important that the software production companies use is stable, secure and innovative. These are also the three fields in where Open Source software is advocated to be very developed.

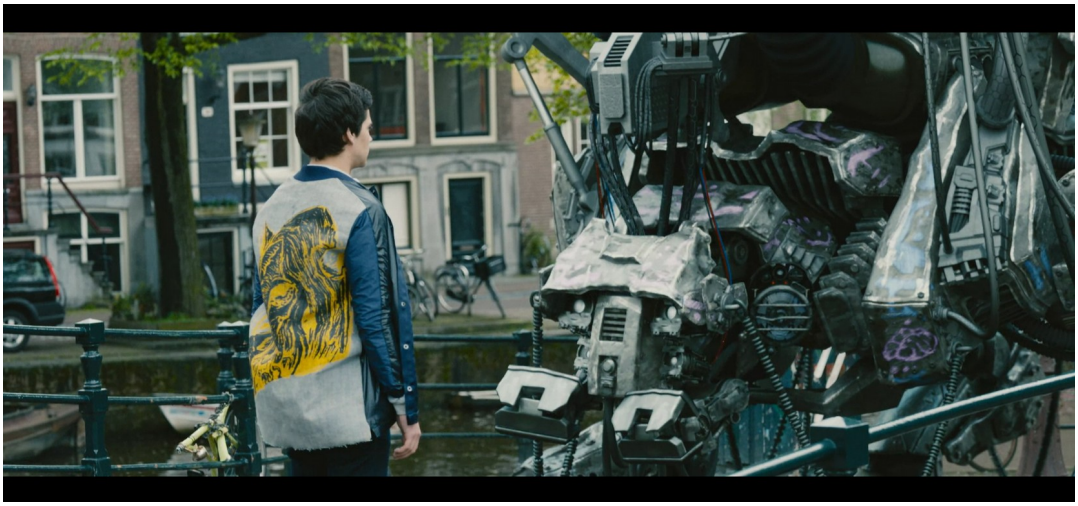


Illustration 2: From the short film tears of steel (2012). All the effects were made with Blender.

Many bigger visual effects[23] houses and film studios in Hollywood are already using Open Source tools. For example Industrial light and magic, Disney, Pixar and Weta digital have

23 In filmmaking, visual effects (abbreviated VFX) are the processes by which imagery is created or manipulated outside the context of a live action shot. Visual effects involve the integration of live-action footage and generated imagery to create environments which look realistic, but would be dangerous, expensive, impractical, or impossible to capture on film.

all been reported to use a GNU/Linux based operating system. The primary reason being that GNU/Linux operating systems are more stable, lighter and further developed than the proprietary alternatives. According to Andy Hendrickson, the director of research and development in California based ILM, a GNU/Linux based operating system: builds distributed computing well, has rock-solid stability, a very low administration cost, and many years of robustness and testing behind it[71].

Many of these bigger studios also use a lot of software that is in house developed and proprietary. These programs are mostly proprietary and protected because of market competition. IE. the studio that creates the better software to make, for example, CGI[24] or visual effects, also has a better chance to get a contract for a film. This can make it very hard for smaller similar companies to establish themselves in the market. If all of this would be Open Source, the playing field would be leveled. The studio that creates the best end product would be leading the marketplace not the one with the best software. Since it is unlikely that the bigger studios would release all their software, smaller companies could still gain an advantage by using Open Source software. Since Open Source software includes the right to modify, smaller studios could build upon existing source code to create their own tools. If they would take it a step further and release their tools under an Open Source license, they would get the full advantage of using Open Source. These advantages include community support, a large developer base, fast bug fixes and a large testing area.

24 Computer-generated imagery (CGI) is the application of computer graphics to create or contribute to images in art, printed media, video games, films, television programs, shorts, commercials, videos, and simulators. The visual scenes may be dynamic or static and may be two-dimensional (2D), though the term "CGI" is most commonly used to refer to 3D computer graphics used for creating scenes or special effects in films and television.

4 The question of ethics

When a user chooses to use Open Source and Free software for the creation of intellectual property, there is an ethical question involved. The main theme of this question is in large part about fair use and moral values. Considering that legally there is no issue involved. The question is: “whether it is fair or morally right to use something that is provided copyright free, to produce copyrighted material?” This can refer to using Free and Open Source software, material released under creative commons licenses, or something released under other Free licenses. Legally the answer for this is yes, because the material or software is released under a Free license.

First of all, there is a psychological and a societal element tied to this question. The psychological element manifesting itself as conscience. Meaning that for some people it does not “feel” right to take something for free and turning it into a profit. The societal element is more of an indication of the society we live in today, because of disbelief in that someone would provide something “no strings attached”. In other words cynicism.

Secondly, there is also a political element, due to the communistic nature of the philosophies and ideas behind free intellectual property. For example even if the Free software Foundation does state that there is nothing wrong with selling software. The freedom a customer then has to share the software is fundamentally anti capitalistic, as an idea. The statement also relies heavily on users having a strong sense of morals and financially supporting the developer of the software. Out of “good will” and fairness, instead of taking the free option. As an idea this is of course very noble but does seem a bit naive. Furthermore, the idea of free intellectual property being a human right is in a way self-contradicting, because it does not respect the rights of the producer of the content.

4.1 Relieving conscience and giving back to the community

There are many ways to relieve and solve the issue of conscience. The most obvious of these being to release all the content one creates as free intellectual property. This does limit one's possibilities in working professionally and leaves the content vulnerable to misuse. A way of protecting the content is using a Copyleft license. For example Creative Commons licenses like “Share Alike” (SA) and “Non-Commercial” (NC). The SA license means that all media that builds upon, derives or remixes the original content must be released under identical terms as the original. The NC license forbids the content to be used for commercial purposes. By combining these licenses a creator can protect their content and ensure that it will always be provided for free.

For a free content creator to support oneself, there are also some solutions. First, a creator can rely on other sources of income than the traditional ones. For example, a film-maker can release their work on a platform like YouTube, that relies on ad based funding. This

way a creator can have some income, while still releasing free content. This is not an ideal solution, as it relies on the popularity of the content to generate funds, but can be combined with other funding methods to support the continuous creation of more content. These other methods will be explored in a later chapter.

Another way to relieve conscience is to fund or give credit to the creator of the intellectual property even if the creator does not specifically ask for it.

Other methods to relieve conscience involve giving back to the community while still retaining a part of the content for commercial release. These methods can be: delayed free release, partial free release, giving back other content, the free release of source material, or the release of unused material.

The delayed free release would work as follows. A creator would generate content and release it under a copyright that would expire after a period. When the time expires, the content would be in the public domain. This is in essence how copyright law works today. The main difference would mostly be in the length of time the content stays copyrighted. The Finnish copyright act states:

"(1) Copyright shall subsist until 70 years have elapsed from the year of the author's death or, in the case of a work referred to in section 6, from the year of death of the last surviving author. Copyright in a cinematographic work shall subsist until 70 years have elapsed from the year of the death of the last of the following to survive: the principal director, the author of the screenplay, the author of the dialogue or the composer of music specifically created for use in the cinematographic work."[72]

In the method I am proposing the time period would be tied to the release of the content and be significantly shorter. A film, for example, might be released as copyrighted and remain that way for the period it is shown in theaters. After this it could be released as free or open film[25] on a streaming platform. Instead of, for example, a DVD/Bluray release, which sales have been declining in recent years. The reason for also releasing the source media of the film would be mainly for educational purposes.

The second method involves releasing part of the content as copyrighted material and part of it free. This is a model that could be viable in game development. In this method a company could create a game that would be released under a Free license, for example the GNU GPL. Then release additional features, bonus content and additional gear, generally known as DLC (Downloadable content), as copyrighted material. A form of this method is already used in the game industry today. Many game companies use a Free-To-Play model. This model gives players access to a significant portion of a game, but players have to pay

25 A term referring to films that are produced and distributed by using Free and Open Source and open content methodologies. Their sources are freely available and the licenses used meet the demands of the Open Source Initiative in terms of freedom.

to get additional content. The difference between my proposed model and the real world model, have to do with the license the content is released under. In my model the main content would be released under a Free or Open Source license instead of the Freemium[26] method used in the real world today. The real world method used today is more of a marketing trick used by companies to promote the content and sometimes blatantly leave out key features of a program in the free version. For example, a save feature. In my model a user has free access to the source code and can thus, develop their own features or purchase features from the provider of the software.

The third method involves releasing some projects as copyrighted and others as free content. This is a fairly straight forward method. A content creator funds their work with copyrighted material and uses part of these funds to create Free or Open Source content. This method can for example be used in 3D modeling. Here's an example, I sometimes use a website called Blendswap to find Creative commons licensed models to speed up my work. These are usually very basic pieces of furniture or generic base human models. I recently noticed that there is no good generic base mesh for a female character, only a male one. So to contribute, I am in the process of modeling and rigging one, that will be released under a creative commons 0 license. Thus, giving something useful to the community. These contributions can be virtually anything, realistic materials, models, lighting setups, etc.

The fourth method, free release of source material, uses a model were only the end result is copyrighted but the material used to produce the end result is under a Free license. The fundamental idea behind this method, is to increase the educational possibilities in a creative field. This method could, for example, be used in the film industry in the following way. A studio releases a film under the standard copyright, at the same time all the footage, effects, plans, storyboards etc. are released under a creative commons (NC) (SA) license in their RAW form. This way a student, or anyone, can have access to the RAW material and study it to learn more about the craft. The reason I have proposed a Non-Commercial and Share Alike license for this material, is to protect it from misuse and exploitation. At the same time the actual film is protected by the standard copyright law. This method does still carry a treat of piracy and this is the reason the footage and other material would only be released in a RAW format. If someone would want to pirate the film, they would have to repeat the whole post-production process. While still adding enough changes to the end result, so that it would legally be viewed as a separate work. After all of this, they would still only be able to release the work under a creative commons (NC) (SA) license, taking away all financial gain. This in itself would at least minimize the treat of piracy.

26 Freemium is a pricing strategy by which a product or service (typically a digital offering or application such as software, media, games or web services) is provided free of charge, but money (premium) is charged for proprietary features, functionality, or virtual goods.

The fifth and last method, is the release of unused material. In this method all the material and the end result would stay under a copyright, with some exceptions. All the material that is not used in the end result would be released under a Free license. In a general design process, a certain feature may go through many iterations and forms before reaching the desired end result. Sometimes this produces a number of material that is never seen in the end result. What I am proposing, is that all this material would be released for public use after the project is finished. In other words, a form of recycling the intellectual property. Another ones trash is another ones treasure, so to say. By just releasing the unused material a designer or company would still benefit from all their copyrighted content, but still give something back to the community.

4.2 Contributing to Open Source and Free software

In addition to contribute back to a community a user can also contribute to the Open Source and Free projects, they used to produce their intellectual property. These contributions are often in the form of source code and bug-fixes, but what if a user is not educated in software development. There are many ways a user can contribute to a project without any coding involved.

The first and most obvious is supporting the software financially. Many Open Source and Free programs are completely driven by donation based funding and this is one of the simplest ways to support a certain software. There is also another reason to fund Open Source and Free projects. This reason is to show software and hardware companies that there is a market place for Free and Open Source software. I stated earlier in this thesis that companies are basically driven by a sole purpose of making money. This also means that they follow and support areas that have a potential market place. By funding an Open Source or Free project users indicate a potential market place in that area. This could, for example, lead to hardware companies increasing support for Open Source and Free software, which can sometimes prove problematic right now.

Other ways to contribute, if financial funding is not possible, is to, for example, help with the parts of the program that do not involve producing code. This can be translating the program or documentation for one's language, participating in the design of the GUI, make artwork for the program, reporting bug’s, teach others about the program, or simply telling people what program was used for the creation of the content. All of these contributions help the program developers, increases the quality of the program and helps build the popularity of the program.

4.3 Building a fair work-flow

To tackle the political aspect of the ethical question, I will in this part address the issue of using an Open Source and Free method for producing intellectual property. This may increase the quality of the product through healthy competition. The reason for this is to minimize the anti capitalistic message of some projects and show that an atmosphere of “fair play” can actually be a viable option, without the negative stigma of communism.

The basis of this method is that when all companies, in a certain field, have access to the same tools and information the competition is more pure. Because the competition is about who makes the best product, not who has the better tools.

This method is based on how a lot of development is done in the Open Source and Free community right now. The number of GNU/Linux distributions listed on Distrowatch.com today is 845 [73]. All of these distributions are in competition with each other for users and supporters, while all having access to the same information. They are all free to borrow solutions from each other and use these solutions to enhance their end product. So the competition is not about who has the best hardware support, access to more funds or a monopoly on an area, but who uses these tools to create a more stable, functional and better designed end product. This makes the competition more fare and healthier. This also makes the collaboration between these distributions easier to solve a common problem. For example, recently the GNU/Linux distribution Solus announced the development of the Brisk menu system[74] for the MATE desktop[75]. The developers of the Ubuntu MATE desktop, another GNU/Linux distribution, expressed enthusiasm toward this project. So they helped fund its development. This is because the Ubuntu MATE project wanted all MATE users to benefit from this menu system instead of just the Solus users.[76] The Ubuntu MATE project and the Solus Project disagree on the fundamentals that make a good operating system, but in this instance they had no issue in working together for the improvement of the product.

This method could also be used in the film industry. In the visual effect's industry many companies create their own proprietary software for making CGI shots. This creates an unfair advantage, because bigger studios have access to more funds. Thus, developing better funded software. This leads to the smaller companies that might be better at their craft, but lack the access to sufficient funds to develop software, being left out of the competition. If all companies would be using Open Source software, all the improvements made to this software would be distributed to everyone. This creates a level playing field with all companies now competing over who can make the best looking CGI shot and not who has the best tools for the job. In Finland. where the resources are limited and the film industry tends to be more small business, the industry could actually benefit from a more collaborative work-flow. Instead of a more competitive one used, for example, in Hollywood today.

The Blender community is another very good example of “healthy competition” and communal sharing of tools. There are numerous tutorials, add-ons, premade 3D models, materials, etc. available for Blender. Blender is a very large and versatile program and to master every aspect of it would take a very long time. But because of the community behind Blender, many tools are made simpler. For example, the primary render engine Cycles is currently lacking in physically based rendering[27], in short PBR. This is supposed to be fixed in the upcoming 2.8 version of Blender. To enable users to improve the quality of their models, many community members have created solutions that they freely share under a creative commons 0 license. This enables users who are lacking in that area to utilize tools, they might not have known how to create. Many of these members still support themselves as 3D-artists, even if they are sharing their methods. The point is that they are not giving away their talent, they are simply sharing their tools.

Another way many Blender users share their tools are tutorials. For example, Andrew Price (aka Blender Guru) has made many tutorials on his work-flows and methods. He is also a founding member of Poliigon, a website that sells textures. In many of his tutorials he shows others how to build material nodes that emulate realism. To be able to make realistic 3D models a user should have access to good textures and know how to apply them. Many of the textures could be created in a program like GIMP fairly easily, but figuring out the correct material node setup, could prove very difficult. The tutorials made by Andrew Price, show users how to build a node setup that emulates realism. These could have been exclusive to Poliigon customers, but they were not. By doing this, Price is not giving away a source of income or his artwork, but sharing the tools he uses to enable others to elevate their skill. This helps grow the user base of Blender, heightens the level of craftsmanship, and helps new users to get a running start.

The nature of the competitiveness in many industries can have a big impact in society. Especially when the competition stands in the way of elevating the craftsmanship in the industry.

4.4 The stigma in Open Source and Free software

Despite being capable of professional quality, Open Source and Free software still carries a certain stigma. Sometimes these programs are thought of as solely for amateurs or non-professional work. The GNU/Linux community is often described as hostile or unfriendly. Free software Foundation leader and figure-head Richard Stallman has been described as too extreme in his views.

Some of these stigmas are based on reality, but many of them are also out dated. The stigma of Open Source and Free software being only for amateurs comes from the times

27 Physically based rendering or PBR is a model in computer graphics that seeks to render graphics in a way that more accurately models the flow of light in the real world.

when Open Source and Free software was not as developed, as it is now. It also has to do with the fact that many of these programs are free (as in free stuff). This means that people who are not working or just starting out in a field have easier access to this software than the proprietary option. This leads to a large part of the community being non-professionals or hobbyists. In film, it all has to do with production value[28]. The thing that is mostly forgotten, is that production value does not come from the software or tools, but from how skilled the craftsman using them is.

Illustration 3: On the left a picture from the film Ozzy (2016), made with Blender. On the right The Secret Life of Pets (2016), made with proprietary software

There are some ways to battle this stigma. First, a user should always advertise to others that they used an Open Source or Free software to create the product. Secondly, if they show their work to others, for example in a portfolio, they should make sure that they keep a high standard and are always trying to become better artists.

The other stigmas of some Open Source and Free communities being unfriendly come from some community members appearing as elitist or hostile. Some of this has to do with members who are very passionate about their views on not using proprietary software. These members often shun members who are using GNU/Linux combined with proprietary software. Other problems come from the anonymous web comment issue. This issue is something that many Internet communities have a problem with, not just Open Source and Free communities. This issue stems from people having a tendency to being more hostile when protected by anonymity.

Another part of this stigma has to do with people being more likely to tell others about bad experiences than good. For example, if someone has a bad experience in a restaurant they are more likely to tell people about it, than when they have a good experience.

This issue does exist and the stigma is based on reality. The thing that is mostly left out is that this problem describes a very small but loud part of the community. Most of the communities are very friendly and respectful. I myself have been using a GNU/Linux distribution for five years and I have never encountered a hostile environment when asking

28 A method, material, or stagecraft skill used in the production of a motion picture or artistic performance; the technical quality of such a method, material, or skill.

for help on forums or just reading forum posts. Many communities have also actively started to battle this hostile behavior with stricter moderation on forums and actively promoting a friendly environment.

Some of this also has to do with many new users not reading the manuals or wiki's[29]. This often leads to community forums having many treads with the same question. A big part of some Open Source and Free communities involve users figuring out things by themselves. For example, Arch Linux is a distribution made for users who like to customize and decide for themselves exactly what software is installed on their system. This sometimes involves a lot of troubleshooting and requires some understanding of how the GNU/Linux Eco system works. To make things easier the Arch Linux distribution has very good documentation on their wiki about what to do when a user runs into problems. When a new user runs into trouble and does not check the wiki or if someone has already asked, but immediately goes on the forum and posts a question. They might get a fairly blunt response. As a contrast the Ubuntu MATE[77] distribution is aimed at new users so a user might not experience the same bluntness when asking for help with something basic.

For the stigma of the Free software Foundation and Richard Stallman there is a very clear reason. It is because their views are fairly extreme. This mainly has to do with them being more of a human rights organization than a practical software advocate, like the Open Source Initiative. This carries with it a certain responsibility to be unwavering in one’s views and principals. If a Free software Foundation member would be supportive of some proprietary software, the credibility of the organization could be questioned.

4.5 Alternate forms of funding

For the production of free intellectual property a content creator still needs to fund their work. A free content creator can but an immense amount of work and time in creating a high quality product for an audience, that might not give anything back for the creators trouble. Often these products are labors of love, instead of hope of financial gain. This is why it is important to support the projects one likes or possibly uses to make the continuous development of this content possible. Nothing in this world is financially free.

To be able to support themselves, free content creators use a number of techniques. These techniques include: donation based funding, ad based funding, support or consultant based revenue, and the development of specialized tools.

The first one, donation based funding, is fairly straight forward. A content creator asks for users to donate money to support the creation of the content. There are many platforms available for this model of funding. These platforms include Kickstarter, Patreon and

29 A wiki is a website that provides collaborative modification of its content and structure directly from the web browser. In a typical wiki, text is written using a simplified markup language and often edited with the help of a rich-text editor. A wiki is run using wiki software, otherwise known as a wiki engine.

Indiegogo. This way of funding has seen an increase of popularity in the recent years and can be a very helpful way of funding a project. It also helps in proving if the project is seen as important to the overall public, because popular projects will have an easier time getting donations.

The second technique, ad based funding, is mostly used by web sites and for example podcast networks. In this method a content creator sells ad space on their platform, and the more popular the platform, the more money the creator makes. This method also works in other ways. For example, the streaming site YouTube bays content creators for driving traffic to their site.

The third method, support or consultant based funding, involves releasing the content for free and selling the training or consultant services for the product. This method is used by the company Red Hat Inc. that make a GNU/Linux distribution. The software is in essence free (as in free stuff) but the user pays for the technical support and training services.

The last method, development of specialized tools, is a bounty based system. It works like this, a company or user puts up a feature request for an Open Source program, to this feature, there is tied a bounty for anyone that develops it. A software developer makes the feature and collects the bounty.

4.6 Industry uses of Open Source and Free software

I already mentioned that many big studios use in house proprietary tools. Sometimes these studios release these tools as Open Source software. The motivations for this is often to create an open standard and to drive innovation forward. Industrial light and magic, Disney and Pixar have all released some Open Source software. In 2003 ILM released a high dynamic range image file format called OpenEXR, in hopes that it would be adopted as an industry standard, because of its Open Source nature. This format received an update in 2014 by a contribution by Dreamworks. Disney and Pixar announced in 2016 that they would be releasing their proprietary software USD[78] as Open Source software. They did this according to Ed Catmull, President of Pixar and Walt Disney Animation Studios, to drive innovation forward in the industry. This could be interpreted as an attempt to attract developers to the software in hopes of improving it.

Some of these releases have yet to bear fruit, as it takes time to develop or integrate code from the more recent technologies. But it is encouraging seeing bigger studios showing interest in serving the field.

In addition to releasing in house software some companies strive to use solely Open Source tools in their pipeline. For practical or ethical reasons. Jupiter Broadcasting, a podcast network based in Washington, has over the years moved more and more tools to Open Source in their audio and video production. Chris Fisher, one of the creators of the Linux

action show, stated in a recent podcast that he first started using Open Source and Free software for the practical advantages, but the more he has read on the philosophy the more he uses these tools for the ethical reasons.

5 The Open Creative Suite

When I started using Open Source and Free programs, I found it fairly hard to find good alternatives for the proprietary programs I had been using. It took me many moths of trying different alternatives before I found the programs I'm using today. Some programs also have a somewhat steep learning curve. For example, when I started using Blender the user interface felt very confusing and complicated. I was eventually able to simplify the interface by using add-ons, tutorials and themes.

The main problem I had with Blender was that it had too many features and I felt somewhat overwhelmed. GIMP (Gnu Image Manipulation Program) could also feel problematic at times. On one hand it felt very easy, because it resembled Adobe Photoshop very much. On the other hand, this is the fact that also makes it very hard sometimes. Because of its resemblance to Photoshop, it can be confusing when features are not in the same place.

To make things easier on others who want to start using Open Source and Free Software, I decided to assemble a software suite. This is the Open Creative Suite.

This suite is a collection of programs that can be used for designing a film or media production. I have made it with pre-production and mainly production design in mind. It consists of: a CAD and a 3D modeling program for designing sets, making construction drawings and creating previsuals; an image manipulation program for editing location photos, creating concept art, and textures for the 3D models; a drawing application for sketching and initial concept ideas; a storyboarding application; a video editing application for editing of animatics and previsuals; and finally, a compositing application that can be used for the design of effects, enhancing animatics or 3D model renders and color correction. The changes and features of these programs will be explored further in the following sections.

The programs I have included are Blender, FreeCAD, GIMP and Krita[79]. These programs are Free and Open Source alternatives to proprietary software such as Maya 3D, 3DS max, AutoCAD, Adobe Photoshop, Adobe illustrator and even Final Cut Pro. There are many benefits in using these Open Source and Free alternatives.

First of all there is versatility, due to the Open Source and Free nature of these programs. Developers of these programs have created a wide variety of features and add-ons or plug-ins that increase the use cases for them. Many of these features have over the years also

been incorporated into the official releases. This makes these programs very versatile and a few of them can replace a wide variety of proprietary software.

Secondly they are very customizable. All the programs included in this suite make their source code available. This means that anyone can download it and compile a custom version for their intended use. This custom version can then for example be uploaded to git hub[30] for other developers to review and improve, leading to a better product. For people who do not know how to program, these programs still offer a high level of customization. Blender for example offers a feature in its User Preferences to completely customize its user interface without any programming involved. Another way to customize these programs is to install changes and themes that others have made. Many of which are released under Open Source, Free or Creative Commons licenses.

The third benefit of these programs is platform agnosticism. This means that they are all available for Windows, Mac or a GNU/Linux desktop operating system. This makes transitioning to these programs easier than many proprietary alternatives, because users do not have to learn a new operating system or buy new hardware for the use of a program. This also means that a user is not locked in to a certain Eco system, for example Apple, but is free to choose the best platform for their use case.

Finally, all of these programs are offered free of price and are funded by a donation based system. This means that a freelancer or starting company, is able to use these programs before actually having any revenue or budget. If they then choose to use and support these programs, they can donate a sum after generating some income to enable the future development of the software.

5.1 The shell script

To make the installation of this suite faster and easier, I have created an installation script that automates the process. This script comes in two forms, a default version and a modified version. Both of them currently work on most GNU/Linux distributions, but development for Mac OS and Windows operating systems is ongoing. The default version installs all the programs with their default user interfaces and commands while the modified version includes changes to the user interfaces that unify their look. Additionally, there are other changes that are separated as individual shell scripts for the default version, but are included in the modified version. There are two different versions of The Open Creative Suite, because I want to provide user with the option to either use the theme I have created and use myself, or use the different applications default themes if they are more used to them. While still, benefiting from a complete software suite.

30 GitHub is a web-based Git or version control repository and Internet hosting service. It offers all the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

The reason for using a shell script that makes changes to the programs after their installation, is a practical one. By implementing all the changes after installation means that the programs are always up to date and their latest versions. This also saves a lot of work for the suite's development, as time is not spent compiling and packaging new versions of the individual programs. By using this work-flow the Open Creative Suites install script remains fairly version agnostic and low maintenance. The drawback of this work-flow is that the changes are not as extensive as they could be if changes were made straight to the source code. On the other hand, to do these more extensive changes would require funding and a team of developers. This project is still in early Beta stages and for now remains a one man operation.

First there is the Blender-mod file. This script adds tree desktop entries to Blender. These are Blender FX, Blender VE and Blender SB. I have already mentioned that Blender is a very versatile program and that it includes functions like 3D modeling, video editing, compositing, animation and a game engine. To make the use of these functions easier for new users I have created three different blend files, with the user interface set up for video editing, compositing and storyboarding. For these files I have made desktop entries so that they have their own startup icons, simulating them as individual programs. The Blender-mod shell script sets all of this up automatically for a new install of Blender.

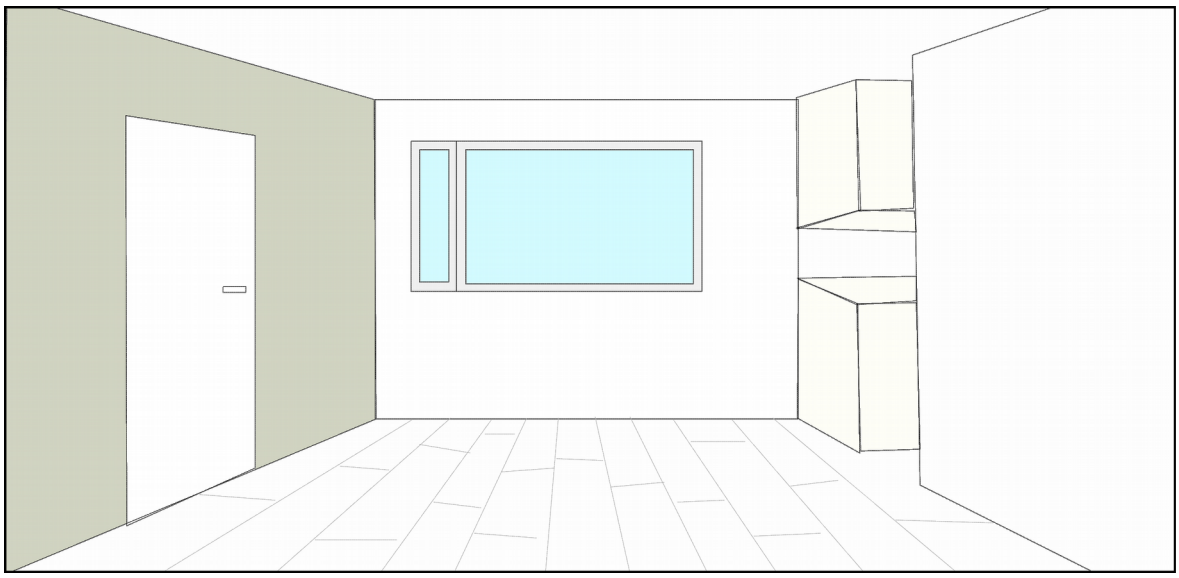


Illustration 4: Simple sketch of a room. Made with Krita.

The second modification script that is included is made for GIMP. Since Adobe Photoshop is generally seen as the industry standard and most people, myself included, that move to GIMP have previously been using Photoshop, I have included a shell script that makes GIMP function and look similar to Adobe Photoshop. This GIMP theme was made by a deviant art user called doctormo and released under the GNU GPL. I have chosen to keep this script separated from the default version and modified version of the suite install

scripts. This is because I do not want to change the default functionality of GIMP and a new user may find it harder to follow a tutorial or course on the program. Instead, the script is included as a separate file and can be installed by those who choose.

Other modifications include a unified theme for the programs that is accomplished by modifying their themes to look similar.

5.2 Included programs

In this segment I have included some information about the programs that are included in the suite. I will not be going in dept on how to use these programs, as a lot of material already exists on that subject in the form of tutorials, books, courses and user manuals. The subjects I will be focusing on are the changes I have made to the programs and why I have chosen these programs to form the suite. I will also include a list (see appendix) and brief description on other useful programs that could be used in a film and media production, but are not suitable for a creative suite aimed for production design. To decide which programs to include I have considered a number of factors: ease of use, platform availability, stability, support and community.

5.2.1 Blender

According to the Blender Foundation Blender is a 3D creation suite which supports the entirety of the 3D pipeline; modeling, rigging, animation, simulation, rendering, compositing and motion tracking; even video editing and game creation. In addition to this, Blender also has a python scripting tool included in the program. This enables advanced users to use Blenders API to create specialized tools.

Illustration 5: Concept art made with GIMP, using CC0 stock photos.

The versatility of this program is exactly why I have included it in the Open creative suite. By including this program I was able to diminish the overall number of programs in the suite. It is also the most advanced Open Source 3D program available today. It has been used to create many short films and even some feature length animated films have been created solely with Blender. To further argue for the inclusion of this program, it is already a widely used program in the film production field.

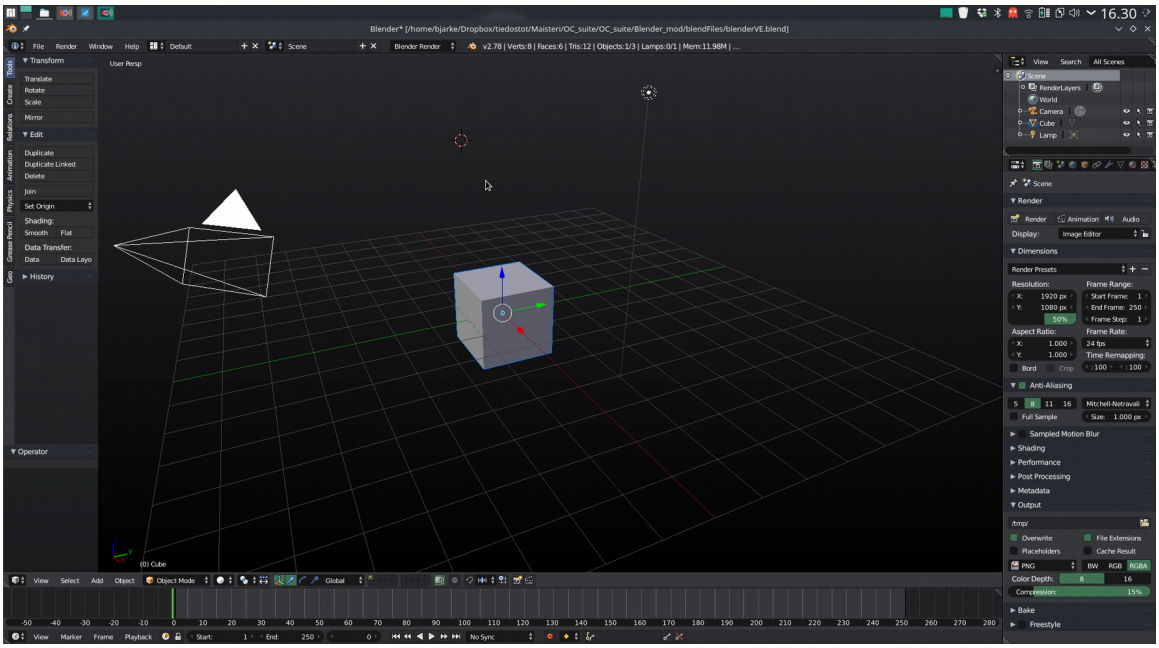
The user interface of Blender is divided into different workspaces. These workspaces are 3D view full, animation, compositing, default/3D modeling, game logic, motion tracking, scripting, UV editing and video editing. In addition to this, users can create custom workspaces. These different workspaces could be thought of as their own programs that are all running on a unified base. This is also why Blender can be thought of as a software suite instead of a single program.

Illustration 6: Concept art made with GIMP

For new users Blender does have a moderately steep learning curve mostly because its number of features. I would advise to treat every workspace as a standalone program and learn them one at a time. For example, first learning to create 3D models in Blender before moving on to the other features such as animation. This also helps speed up the learning process as it familiarizes the user with the basics of the user interface. Blenders work-flow is also heavily based on using keyboard shortcuts and just learning these helps speed up the actual work. To make things easier for Maya and 3DS max users the developers have included setting that convert all the keyboard shortcuts and mouse control to use the ones

found in Maya or 3DS max. For new users I would advise against using these settings, because it might make following tutorials and manuals confusing.

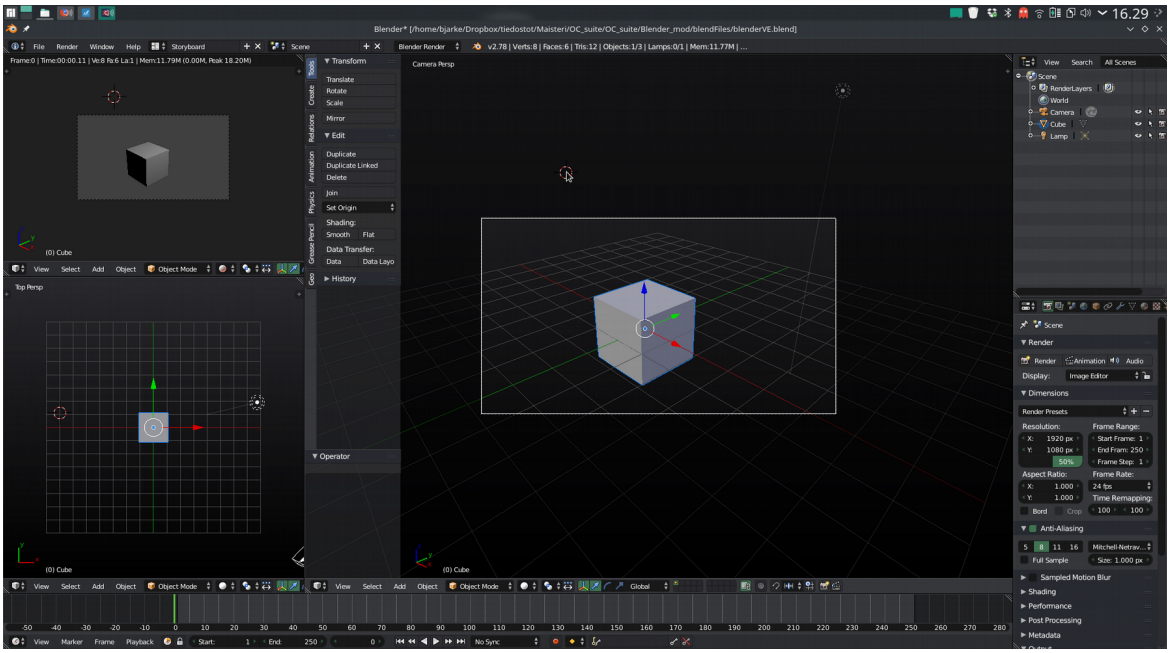
I already mentioned that I have separated Blender into four desktop entries: default Blender, BlenderFX, BlenderVE and BlenderSB. This is solely done to make the different workspaces easier to comprehend, because they are treated as their own programs. Blender does include more workspaces, but I have chosen not to separate all of them. There are two reasons for this. First, some workspaces are features that in my opinion, should be included in a 3D modeling program. For example, 3D modeling, animation and motion tracking could all be thought of as features in a single 3D application. Secondly, The Open Creative Suite is currently aimed at doing production design, so the game logic workspace is not separated for it is not relevant to production design. All of these features are still included in the default Blender entry, even if they are not made into their own desktop entries.



The default Blender entry is in The Open Creative Suite representing a 3D modeling application. If the user installed the default version of The Open Creative Suite, they will be presented with the default workspace of Blender when they launch the program. However, if they have installed the modified version of The Open Creative Suite, they will be presented with a customized workspace for 3D modeling. I have done this because many new users find Blenders default workspace view overwhelming or confusing. To simplify the workspace I have first increased the 3D view's size and made the tool panels transparent, to maximize the 3D view. I have also made the overall theme darker. This is done to simplify the workspace and make it easier to look at, for long periods of time.

The BlenderSB desktop entry is a custom workspace that is optimized to create storyboards. It is based on a work-flow used in the animated short film *Cosmos Laundromat (*2015) by storyboard artist Matias Mendiola. This work-flow utilizes the

Grease pencil tool to draw storyboards inside a 3D view. The tool was initially developed to quickly make hand written notes in the view port, but has since grown in to a sketching tool.

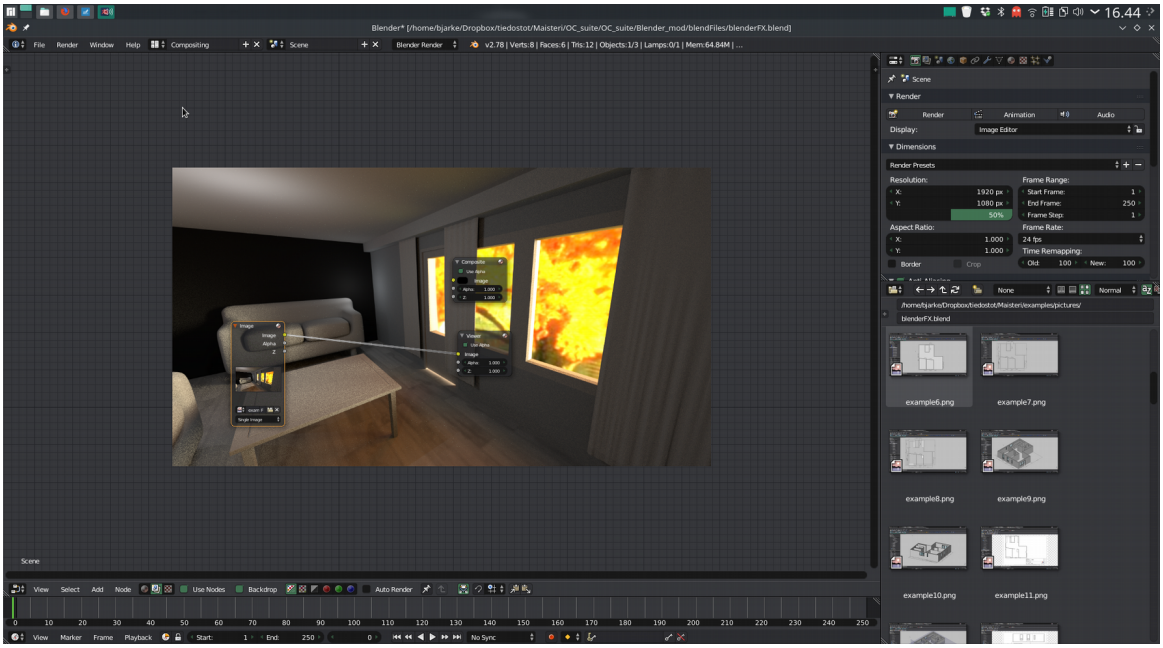


The workspace is built out of six panels: Main view, secondary view, top view, time-line, outliner and properties. I would also advice the use of a Wacom style drawing tablet or touchscreen for a more natural drawing experience and fully utilize the Grease pencil tool. The BlenderSB workspace is also set up so that users who do not want to use the Grease pencil tool can instead use 3D models. To make this simpler I have included a male and a female model in the BlenderSB startup file. These are pre-rigged and fully posable.

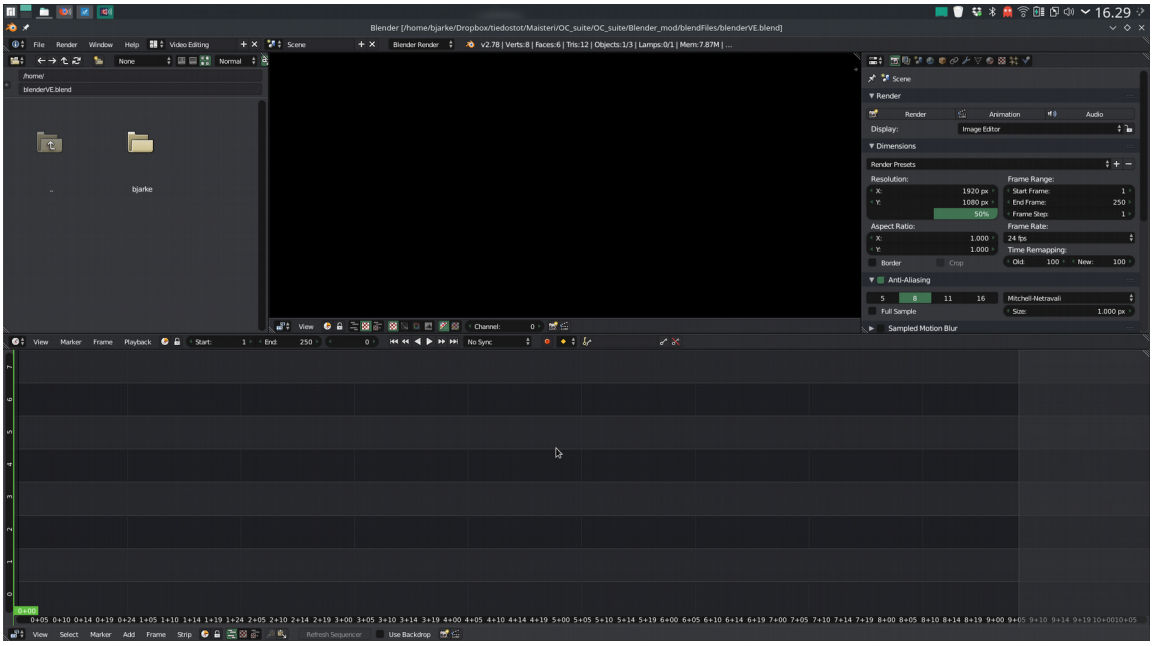
The panels in the workspace are setup in the following way. On the left side there are two view-ports, secondary view and top view. The secondary view is there so that a user can use a different view-port shading than in the main view, like for instance Rendered view. This shows a user how the picture will look when it is rendered out. The top view is there for setting up the scene. This way a user can for instance have a floor plan on the bottom and position all the lights, camera and furniture accordingly. In the middle of the workspace there is the main view. This is where most of the actual drawing will be done. On the left side of the main view a user will also find the options for the Grease pencil tool. This option panel can be hidden or visible by pressing the letter **T.** On the right side of the workspace are located the outliner and the properties panels. The outliner is there to keep track of all the objects in the scene and the properties panel can be used to set the resolution, frame rate, aspect ratio and other settings. Finally, on the bottom of the workspace there is a time-line panel for adding key frames and animation.

BlenderFX is a desktop entry made for the compositing workspace. The reasons I have included it in a software suite aimed at production design, is because it can be used to add

detail and enhance videos of the design models. It is also a useful workspace for working on previsuals and animatics. These videos and previsuals are a very useful way to showcase the sets and locations for a film. The compositing workspace in Blender uses a node-based compositing engine similar to for example The Foundries Nuke[80].



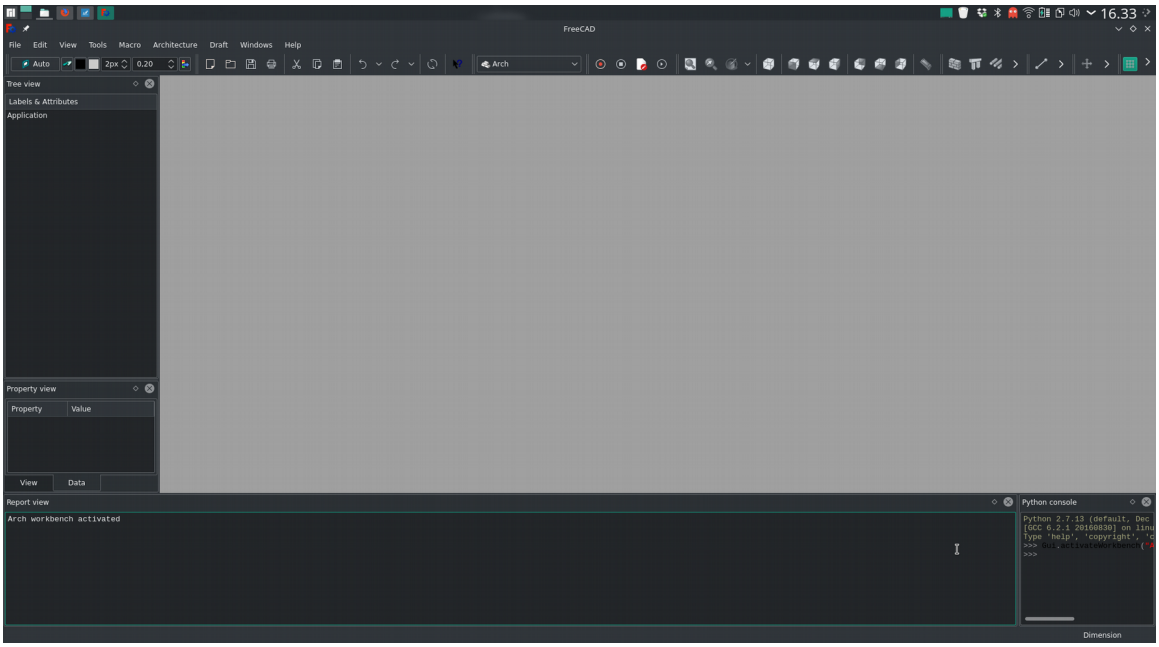
When a user launches BlenderFX, they are presented with a simplified version of the default compositing workspace. It is composed out of four panels: node editor, time-line, file browser and the properties panel.



The Blender VE desktop entry opens a modified version, similarly to BlenderFX, of the video editing workspace. I have included it in The Open Creative Suite so that users can edit together animations created of their models. The video editing workspace found in

Blender is one of the few Open Source and Free video editing tools, that are suitable for professional work. Open Source and Free video editing software is not very commonly used in the professional field, but Blender has been used for editing a feature length film. The workspace is set up to mimic Final Cut pro, because of its popularity and clear work-flow.

5.2.2 FreeCad



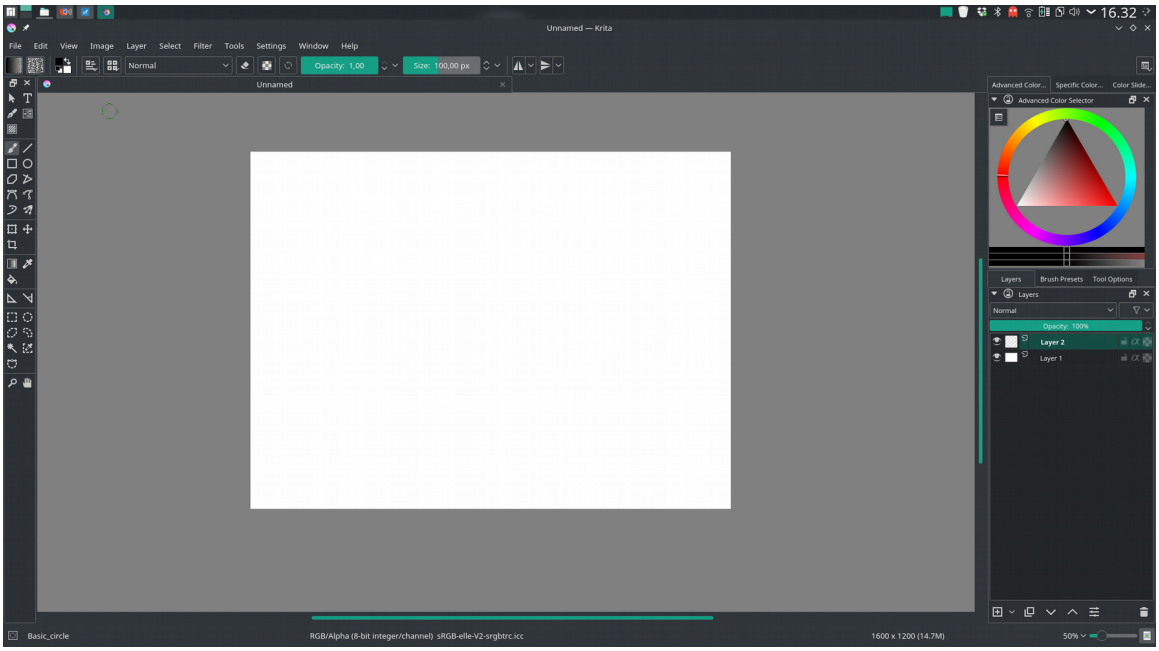
FreeCad is an Open Source and Free computer-aided design (from here on CAD) program. The reason I have chosen it for the Open Creative Suite is because it has more suitable tools for production designers than many other Open Source and Free CAD programs. Similarly, to Blender it uses different workspaces for different purposes. The most important workspaces for production designers are the arc, the drawing and the draft workspace. The arc workspace is originally made for architectural design but is equally suited for designing studio sets. The draft workspace is made for 2d design and is useful in creating floor plans. The drawing workspace can be used to generate 2d plans of a 3D model.

In The Open Creative Suite I have decided to keep the changes to a minimal. This is because FreeCAD has a relatively steep learning curve and can sometimes be confusing for new users. To make it easier to learn I have included a couple of tutorials that helped me get started with the program. These tutorials are provided by the FreeCAD developers.

This program can be used for creating build plans, 3D models, floor plans and smaller components that can be 3D printed or machined. The last feature could for example be used in creating special props or miniature models. The difference with FreeCAD and Blender is that Blender is more aimed at design and uses a more organic work-flow, while

FreeCAD is aimed at engineering and offers more precision. For production design I would recommend using Blender when creating the design as well as the concept material and FreeCAD for the actual build plans of the studio sets and props.

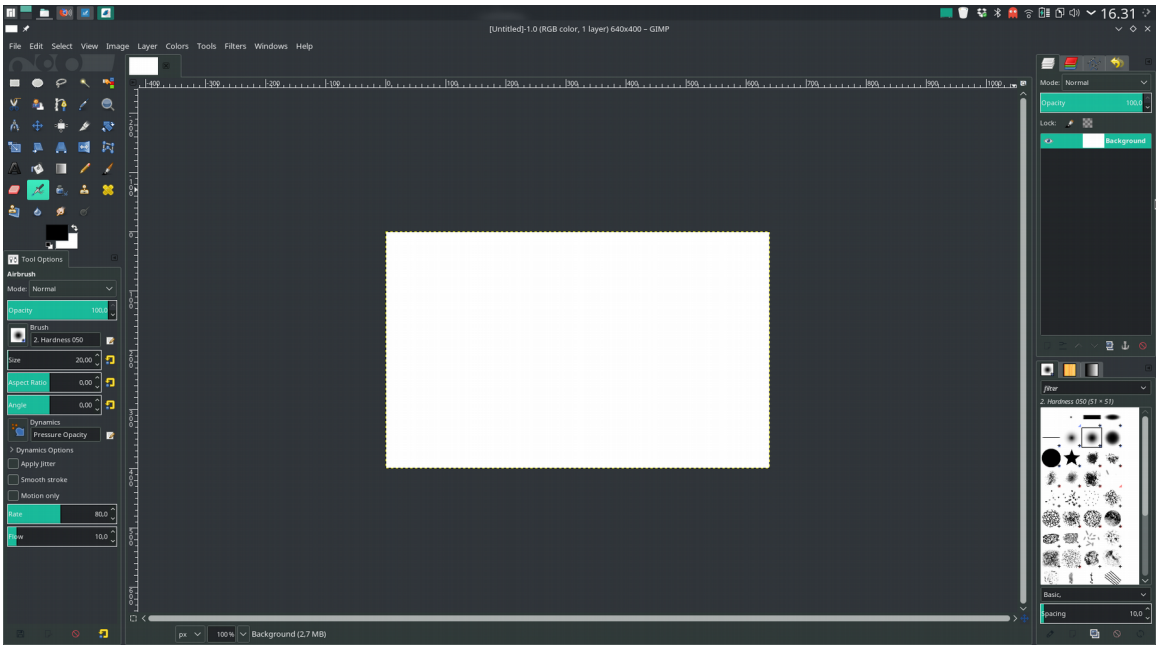
5.2.3 Krita



Krita is a digital painting application similar to Adobe illustrator. It was originally created for image manipulation and digital painting but the developers decided to focus solely on painting in 2009[81]. It is a very competent tool for sketching and concept art. It can also be used for more traditional storyboarding than BlenderSB. The work-flow of Krita is very similar to Adobe's software so users who are used to Photoshop and Illustrator should find it very easy to use. The Krita foundation offers three different versions of the software, Krita Desktop, Krita Gemini and Krita Studio. Krita desktop is the basic version of Krita. Krita Gemini is a version of the program that is designed to use with convertible and tablet computers. It is for now only available through Valve's Steam store. Krita Studio used to be a payed version of Krita, which included commercial support. The Krita Foundation has changed this to instead offer the commercial support through support subscriptions and dedicated development. I have decided to add the desktop version of Krita to the Open creative suite for it does not require the steam store. I have also created a customized theme for Krita to unify the look with the rest of the suite. The Krita User interface is relatively simple and easy to use. Because of this, I have not made any customization to the functionality. For this program I would recommend the use of a Wacom style tablet.

5.2.4 GIMP

GIMP or Gnu Image Manipulation Program is the Open Source equivalent to Adobe Photoshop. The development for GIMP started in 1995 and has for many years been the default Open Source alternative to Adobes proprietary Photoshop. GIMP also has a very similar functionality to Photoshop but can be difficult due to having different commands, icons and names for its tools. To make things simpler I have added the modification shell script, which I described earlier. The theme has also been modified to resemble the rest of the suite.



The reasons for including GIMP in the Open creative suite are its stability, community size, age and image manipulation properties. It is a very powerful tool when, for example, modifying images of real locations. Due to its fairly large user base and community, tutorials, support and additional plug-ins, are available in mass.

5.3 Other programs and future development

This project is ongoing and I will keep developing it further in the future. Plans for future development include a more unified look and functionality for the programs, which can be accomplished after the developers of GIMP and Blender finish their future versions and changes. The GIMP project is currently transitioning its user interface development to use Qt5, this will make changing the user interface look and functionality easier. To make changes now would be futile, because these changes would have to be re-implemented after the move. Blender is also currently in a process to separate their user interface from the underlying source code. In older versions of Blender the user interface design has been embedded in the C/C++ code for the main application. Newer versions of Blender have moved parts of the user interface to separate Python scripts. When the move is finished, the

user interface development will be much simpler. Thus, enabling the further development of a unified user interface for the Open Creative Suite.

The expansion of this suite into a full film production suite is also an option. This would require the addition of programs suitable for script writing, production, color correction, audio production and such. This could also be solved by creating a GNU/Linux distribution with all the software included.

The best software I have personally found for writing screenplays are Celtx[82] and Trelby[83]. Celtx is a very popular program that has many useful features for writing screenplays. The problem with this software and why I personally prefer Trelby, is that the Celtx developers decided to focus all their efforts on the web only version. This version uses subscription based funding. The web based version can not truly be called Open Source and Free software as it uses a proprietary server component. Furthermore, the last update for the Open Source desktop version of Celtx was released in 2012. Similarly, the last stable version of Trelby was released in 2012 but does seem to have some light development done to it from time to time. It is also a very functional program that still works today and I have found it adequate for my needs.

For a producer the Open Source community offers very stable and well-rounded tools. There are office suites like Apache Open Office[84] and LibreOffice[85], LibreOffice being the more popular one. Both of these programs offer many of the same tools found in Microsoft Office[86] and are very capable for processing documents. Other Open Source and Free software, a producer might find useful, are messaging applications like Telegram[87]. This program is available for Android, iOS, Windows, Mac OS and GNU/Linux. It specializes in text chat and file exchange. For conference calls a software called Mumble[88] could be used instead of Skype[89]. For cloud storage there are Nextcloud[90] and ownCloud[91] that both have similar features to Dropbox[92] and Google drive[93].

For the director of photography (form here on DOP), the Open Source community offers some useful tools. There is the already mentioned GIMP for image manipulation and Krita or Blender for storyboards. For monitor calibration there are tools like DispcalGUI[94] and Gnome color manager[95]. I do not have any experience with DispcalGUI but it should have more features than Gnome color manager for professional users. For color correction I have not found many specialized tools for film. Darktable[96] is a color correction raw image editor, but it specializes in images not film. Other tools that can be useful for the job are Open Source and Free video editors. For example, the Blender compositor offers a color correction node and Kdenlive[31] [97] does have some features for the task. There are also users who have written scripts to enable video color correction in Darktable, but these are not supported by the official development team. Many tools are in development for

31 Kdenlive is an Open Source and Free video editing program.

film photography and this field may experience growth in the future. For example, the Apertus project, which is striving to create Open Source and Free digital cinema tools for filmmakers. This project is crowd funded and has received an EU innovation grant.

Audio production is a field which has received a lot of development from the Open Source and Free community. There are many tools available for this field. A good source for more in dept information on this subject is Libre music production, which is a community driven online resource. Some basic useful tools include Audacity[98], Ardour[99], Rosegarden[100], Lmms[101] and Qtractor[102]. Audacity, Ardour and Qtractor are all programs for editing and recording audio. Lmms and Rosegarden can be used for the creation of music, for example film scores. There are also many services that offer creative commons, royalty free and public domain sounds and audio bites. Some of these are web-sites like Audioblocks, Sounbible and Freesound. Other useful tools could also be Free and Open Source video editing tools that will be covered in the following paragraph.

Video editing is a field that still seems to be problematic in the Open Source and Free community. There are many programs available and that are in current development. The main issue is that there are very few that are stable and good enough for professional work. Many programs also have a user interface that can seem complicated for users that are used to Final cut pro, Adobe Premiere and Avid. Blender has a very capable and most important stable video editing workspace, but it can have a steep learning curve. Especially for users who are migrating from industry standards. Another tool that has been improved lately is Kdenlive. This program is very promising and might become a pro tool in the immediate future. Lighworks[103] is a video editor that is aimed at professional editors and it has been used in films like *The Wolf of Wall Street (2013)*, *Pulp Fiction (1994)* and *The Kings Speech (2010).* The problem with this program is that it is still proprietary software. The developers of Lightworks announced that they will release the source code for their software in 2010, but before this release the software would first be released for GNU/Linux as well as Mac and the code would be cleaned up. Six years later the program has not been released. If Lightworks does release their program in the near future, there will be a period of immediate improvements in this field. For now, I recommend learning and using Blender, because it is already used in the professional field.

Lastly there are many Gnu/Linux distributions for those who want to work on a Free and Open Source platform. The ones I would recommend are Ubuntu, Ubuntu Mate, Ubuntu Studio[104], Elementary OS[105], Redhat, CentOS and Arch Linux. Ubuntu is a very popular distribution and offers a large community for support and a very stable platform. It also works without issues on most computers, even Macbooks. Ubuntu Mate is a version of Ubuntu with the Mate desktop instead of Unity and my personal favorite. This distribution was originally founded by Martin Wimpress and Alan Pope. It is very functional and utilitarian in its design, with simplicity and ease of use in mind. Because it is based on Ubuntu is shares all the benefits that Ubuntu offers while leaving out many of the

disadvantages. Ubuntu studio is a GNU/Linux distribution aimed at media production. This means that it comes with many programs for the creative field preinstalled and some custom configurations. It also uses a different desktop than basic Ubuntu called Xfce[106]. This desktop is very light and should be very easy to use when migrating from Windows. Elementary OS is a distribution made especially for Mac OS users. It is also based on Ubuntu but includes many changes and customization's. The developers of this distribution have a very strict design policy and they strive for simplicity, consistency and stability in the user interface. This is a very good distribution for Mac OS users who want to migrate to an Open Source and Free platform.

For users who want company support and do not want to spend time looking for solutions to problems, the Open Source and Free community offers Redhat. This is a GNU/Linux distribution made for enterprise use by Red Hat Inc. This company differs from many other community driven projects, because it is not based on crowd or donation based funding. This is a commercial company. The difference between Red Hat Inc. and for example Microsoft is that Red Hat still offers an Open Source product and revenue is generated by offering support and training courses. A user who does not feel the need for this paid support and help services can get the non paid version of Redhat in the form of CentOS. There is also another version like this called Fedora, but this distribution is more of a testing platform for features that may later be incorporated into Redhat. Due to this, it has a very rapid development cycle, with a new version coming out every six months. This is not necessarily a bad thing, because it usually has the newest technologies, but this may lead to stability issues.

Arch is a distribution that offers complete control. When a user installs this distribution they will only get a base system that they can then build upon, deciding themselves which desktop environment and other packages and programs they need. This method of building a distribution has its advantages and disadvantages. For example, a company or user can build a very lightweight and simple operating system that only has the essentials for their intended use. This distribution is also a so called rolling release, which means it does not release new versions of the system that need to be updated and reinstalled. Instead, the base operating system is constantly updated and a user will never need to upgrade it. The disadvantages in this distribution is that a user needs to have some understanding of how GNU/Linux operating systems and computers work. They also need to be able to be their own tech support. This is not a distribution for new users. For users who want to learn about this distribution the Arch community offers very good and in dept documentation, tutorials and wiki pages.

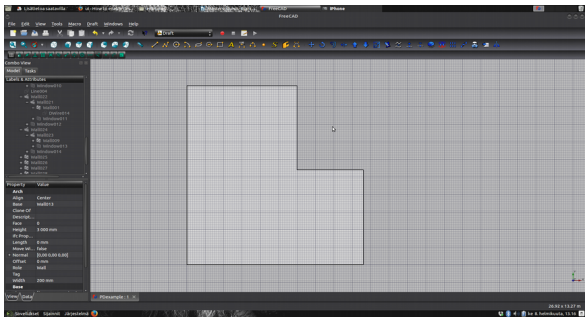
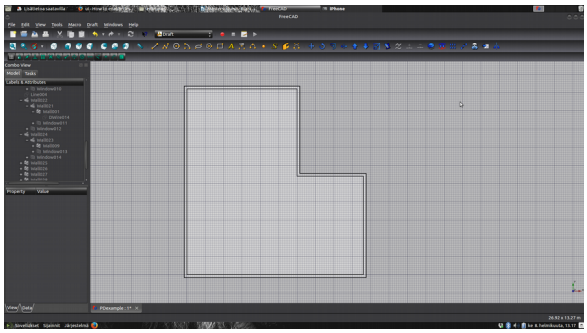
For those interested in these different programs I have compiled a list of Open Source and Free software for film and media production (see Appendix). This list can be found in the attachments for this thesis.

5.4 Work-flow example

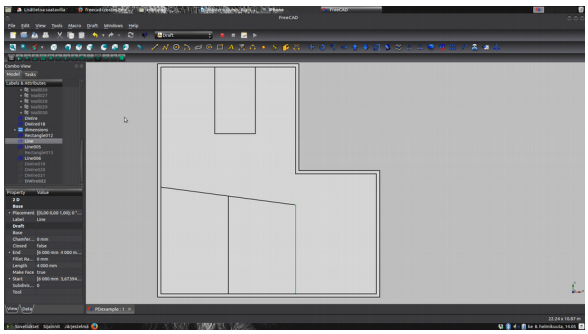
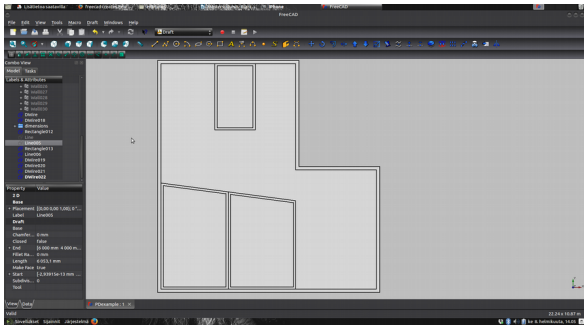
To showcase how The Open Creative Suite can be used, I have made a work-flow example. In this example I will go through all the programs in the suite and show how they can be used in a real world scenario. The premise is to design a studio set of a real apartment, for an imaginary show. For this I will create a 3D model and build plans using Blender and FreeCAD. I will also showcase the storyboard and previsual/animatic tools by staging a short scene.

5.4.1 3D model and build plans

For making the 3D model of the apartment, I would recommend starting with FreeCAD. This will make the model more precise and it can be used to generate build plans later. The simplest way to start is using the drafting workspace to create a floor plan. The first thing to do is using the Line, Rectangle and Draftwire tools to create guidelines. To start I make an outline of the apartment using the Draftwire tool. I recommend starting from the 0,0,0 coordinate in the x,y,z space. Then I use the Offset tool to make a copy of this shape that is 150 mm larger than the original on all sides. These two shapes will work as guidelines for drawing the outer walls of the apartment.

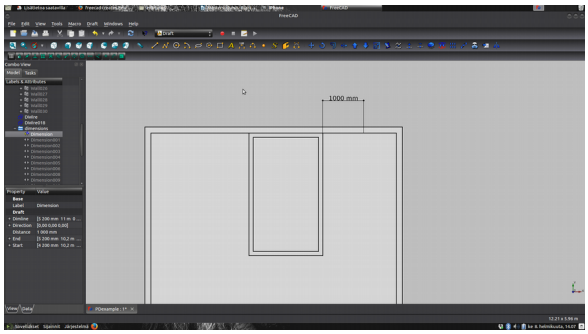



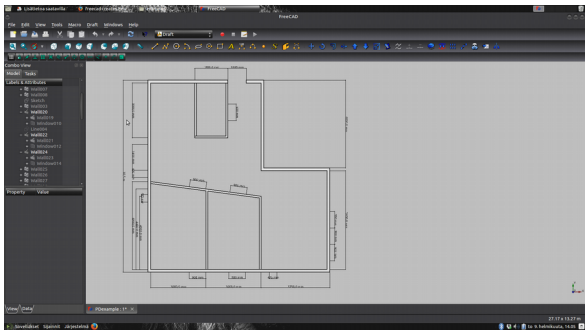
I now use the same technique to make the inner rooms of the apartment. For the two rooms that are next to each other I use the line tool to draw the slanted wall and the Draftwire tool to make two closed shapes of the individual rooms.

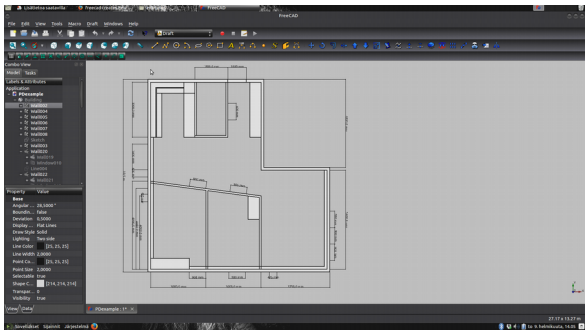
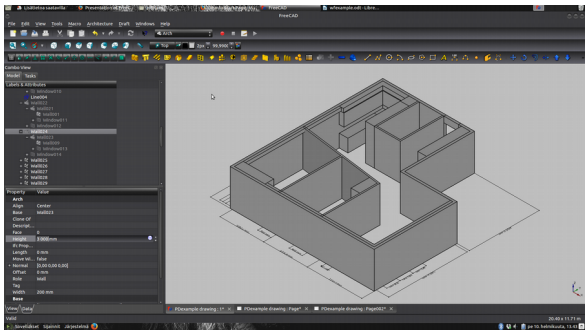
Next I put in a measurement of the outer door's width in the upper right corner of the apartment, using the Dimension tool. After this, I have two lines intersecting with the outer wall guideline shapes. These guidelines can now be used with the Draftwire tool to draw

one solid shape to represent the outer wall. This solid shape will later be used to create the actual 3D wall. I repeat this step to make the dividing walls for the rooms.

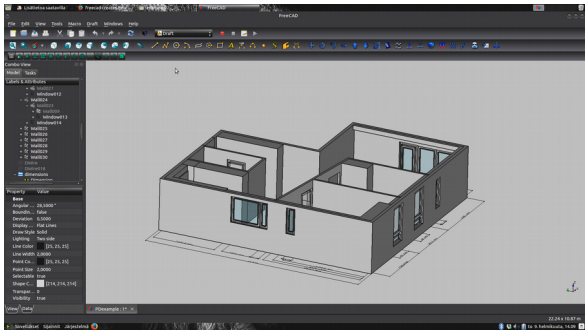
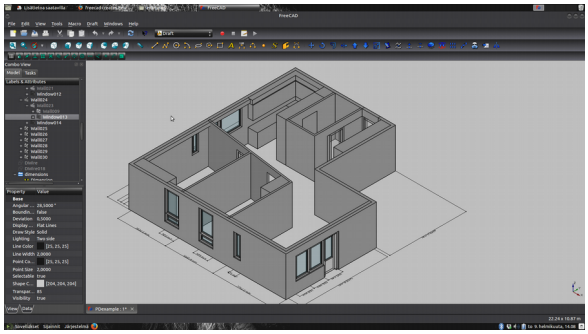




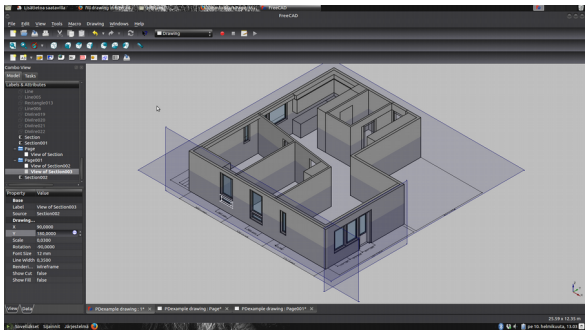
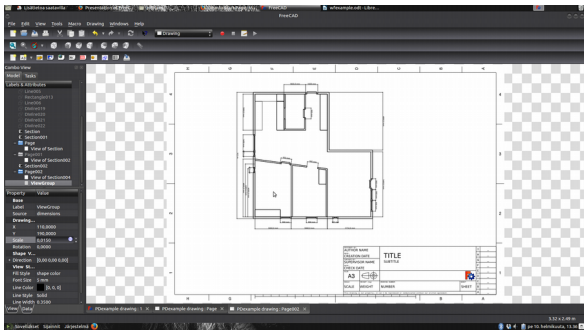
Before I start working on the 3D shapes, I make rectangles to represent the cabinets in the apartment and add all the dimensions for the windows, doors and walls. After this, it is time to change to the Arc workspace.

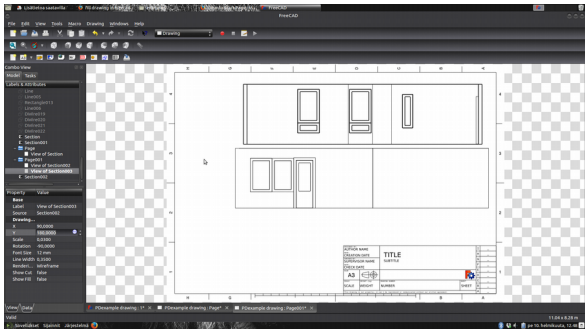
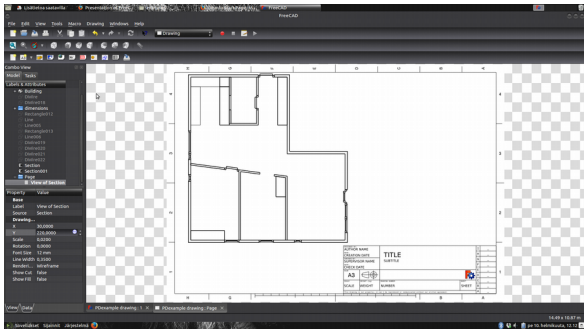
Making walls in FreeCAD is made very simple when using this work-flow. This is mainly because any 2D shape can be turned into a wall. To make all the walls and cabinets I select all the rectangles and shapes made with the Draftwire tool and simply press the wall tool. This tool automatically converts the shapes into walls that are three meters high, so a user merely needs to adjust the height and Z position if need be.

To add the windows and doors to the model I use the window tool. This tools lets a user make a variety of windows and doors, either using the presets or a custom sketch. FreeCad does not require a user to make openings for these, but automatically calculates this for them. It will also resize the opening when a user resizes or re-positions the window or door. After adjusting all the walls and other shapes and adding all the windows and doors, I move on to generate the 2D construction drawings.

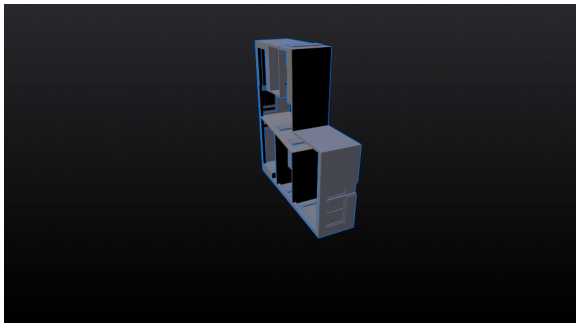
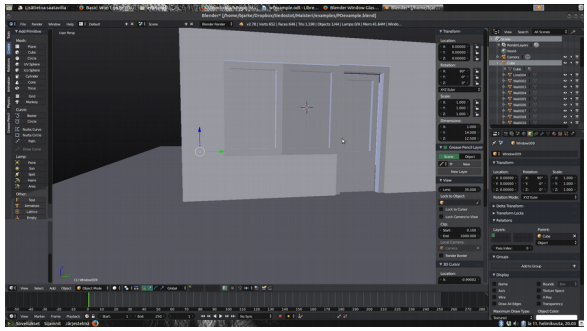



To start working on the construction drawings I add section planes to the model. These section planes are very useful when a user wants to generate construction drawings of specific objects. To these section planes I can now add the objects I want it to display when generating a drawing. To work on the drawings I change to the Drawing workspace.

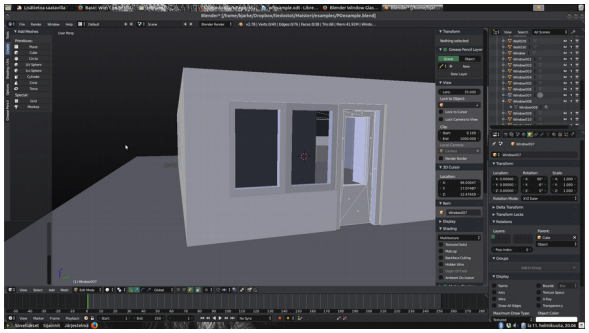
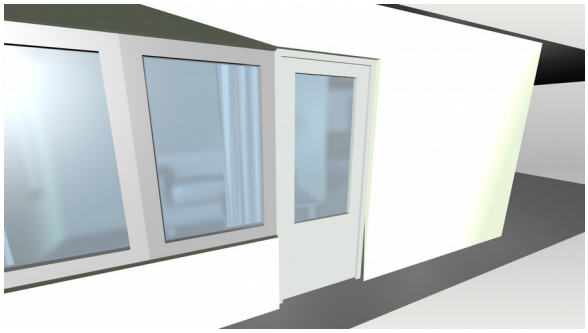



In this workspace a user can use presets or custom templates of papers to display the drawings on. To generate a drawing I select one of the section planes I recently created and use the Draft view tool. This tool generates a 2D image of the objects in the section plane. After this I adjust the scale and position of the image. The generated image can now be exported from FreeCAD as a PDF file. The section plane method is just one out of many to generate construction drawings. For example, when adding the dimensions I created earlier in the draft workspace. I can group them together, select the group, and use the draft view tool. After that I can position and scale them to correspond with the drawing generated with the section plane.

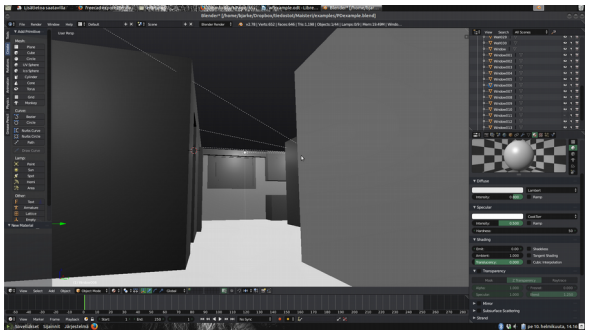
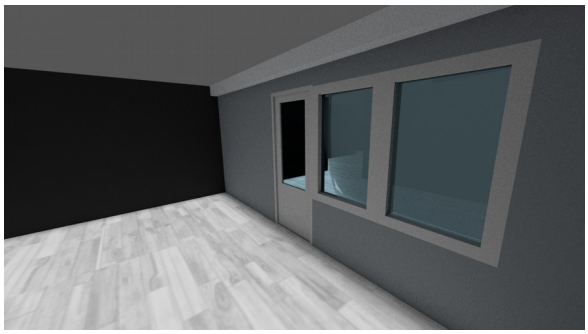
When this is done it is time to export the 3D model and import it to Blender, for assigning materials and creating concept art. It is also possible to assign materials in FreeCAD, but Blender's render engine Cycles is more capable. The format to export can be .stl or .obj (wavefront).

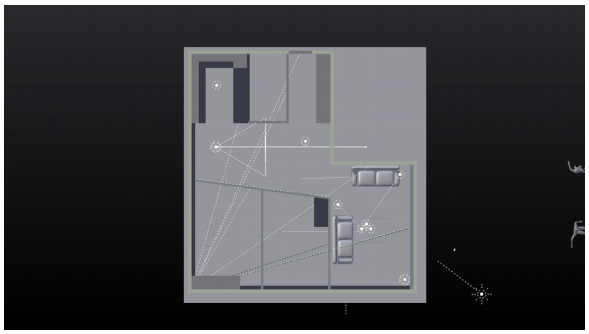
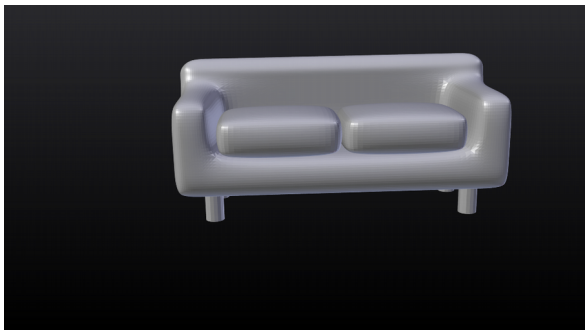
When importing the model to Blender, there are some things that need to be adjusted. First of all Blender uses three types of units: imperial, metric, and its own Blender units. This can sometimes lead to the model appearing larger than it should be, so the scale needs to be adjusted. When importing in obj. format, I have also come across the model being rotated to the wrong angle. After adjusting these, it is time to refine the model.

I start by fixing all the windows, because these were imported in obj. format they are no longer displaying as transparent. As a very fast fix I can just enter *edit mode* in Blender and delete the faces that represent the glass. For a more realistic window I can also replace the face with a plane and assign a glass material. I then proceed to make small adjustments to the doors and door frames. To be able to texture the model correctly I also separate the walls into separate objects.

When all the adjustments have been made, I can move on to create concept art. I also save a copy of the model, because it is now in its base form and ready to be edited or used for storyboards, animatics or lighting tests.

To make the concept art, I start with adding furniture. To make this faster I use a furniture database, which I have compiled from models found on the web, with creative commons

zero licenses. For example, there are web sites like Blendswap where members of the website can upload models for others to download. It works with the Open Source principal that one member uploads a model and others can improve upon it. It is not mandatory to upload anything, but it would be “good practice”. For my furniture database I only use creative commons zero licenses, because they provide the most freedom.




When the model has been furnished and walls have color I can now create concept art. Blender has two render engines: internal and Cycles. I recommend using Cycles, because it offers more options and features. The Blender developers are also more focused on it. For the concept art I can for example show the apartment in different lightings, seasons or colors. These rendered images can also be modified in Blender with the Compositing workspace by using nodes, or they can be imported to GIMP for modification and refinement.

5.4.2 Materials and texturing

One of the most important aspects of making realistic 3D models are materials and textures. To make realistic materials in a CGI environment a user should be able to create textures in GIMP and use the material nodes in Blender. Some of the most important things to take into account when making textures is fresnel, imperfection, and dirt. All three of these things can be found in almost everything in the real world. First of all fresnel. The term fresnel refers to an effect, found in every material in the world. This effect, is that everything reflects light, when viewed from a certain angle. S*ee illustration 7.* In this picture there is a porcelain coffee cup. It has a fairly rough material. When light hits the cup from the zero angle it is not reflective, but the closer the light angle gets to 90 degrees

the more reflective the surface becomes. This is the fresnel effect. To make objects appear more realistic in a CGI environment, this effect needs to be replicated.

Illustration 7: An example of the fresnel effect, the cup is more reflective near the edges.

The second thing a model needs to appear realistic is imperfection. One of the biggest things that make CGI models look unrealistic is that they are too perfect. All edges are perfectly square, all colors are one uniform color, everything is perfectly symmetrical, etc. This is why a user needs to add imperfections to the models. What this means in material and texture creation is the addition of color variation, scratches and dings. Every item in the world has some measure of these.

The third thing is dirt. The world is a very dirty place and to make a model appear realistic a user needs to add dust, fingerprints, wiping residue, etc.

One of the biggest reasons I included GIMP in this software suite was to make the creation of these materials and textures easier. A basic CGI material usually contains these texture layers: Diffuse map, Normal map or Bump map, and Gloss map. GIMP includes many tools and plug-ins to simply create these texture maps. I have also included the GIMP-plugin-registry in the Open creative suite installation, because it contains many plug-ins required for 3D texture generation. By using GIMP's texture generators coupled with the PBR material shader nodes, provided by the programs communities under a CC0 license, a user can make a 3D model appear very realistic.

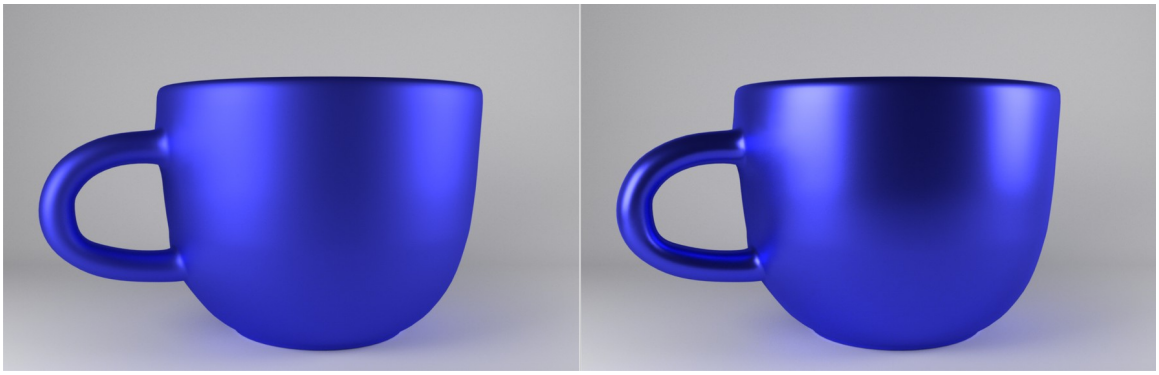


Illustration 8: A 3D render of a cup made in Blender. The left one is using a default Diffuse material. The one on the right is using the default fresnel node added to the Diffuse.

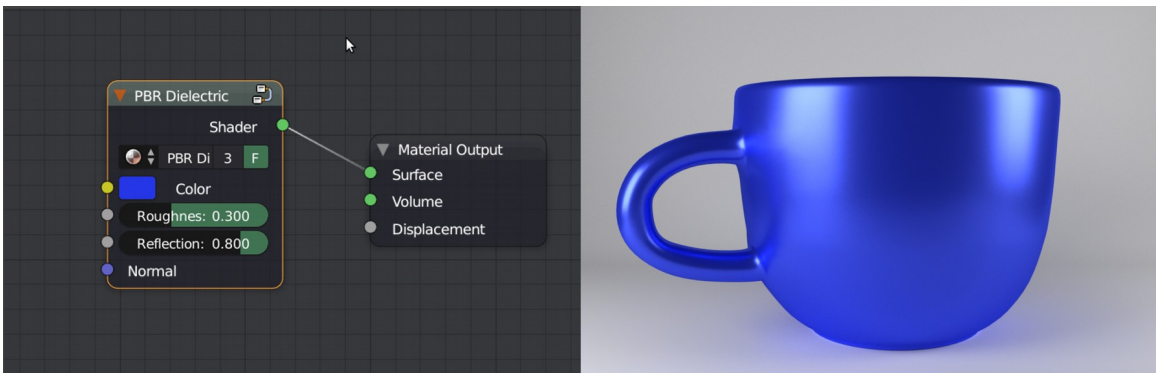


Illustration 9: The cup with Andrew Price's PBR Dielectric fresnel node.

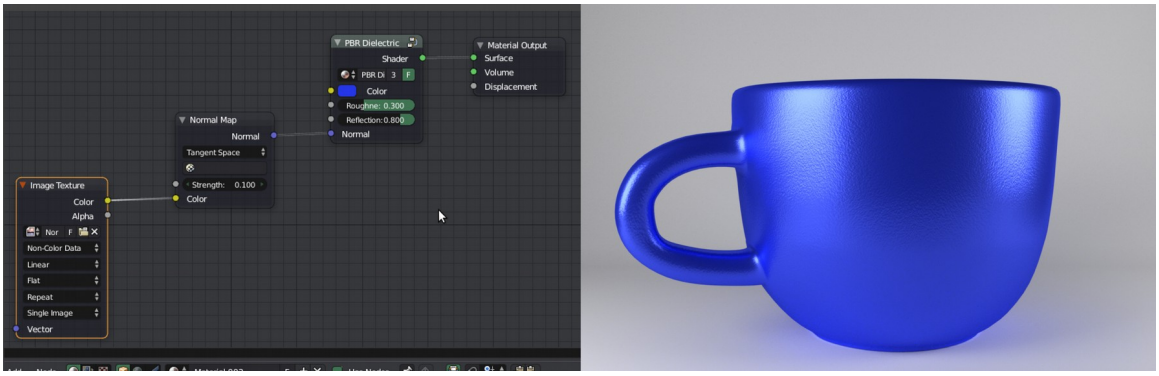


Illustration 10: The PBR dielectric node with added Normal map, generated in GIMP

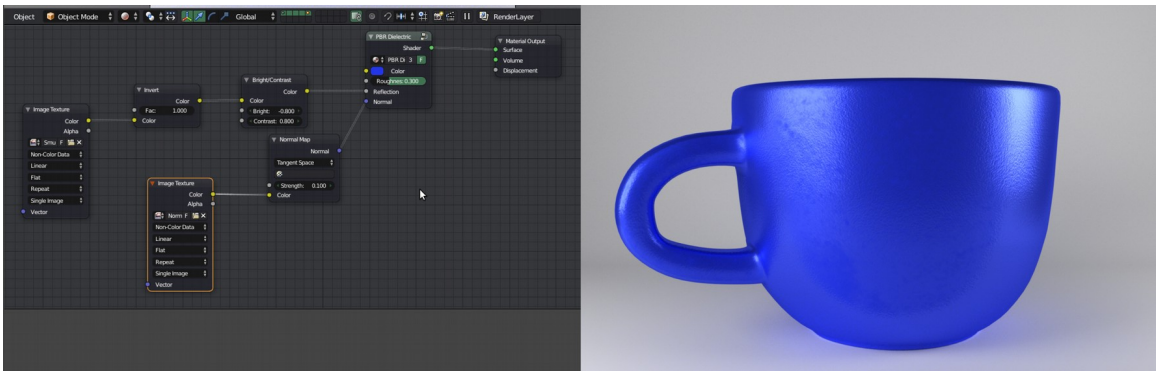


Illustration 11: Same node setup with additional smudge. Added with a Gloss map, made in GIMP.

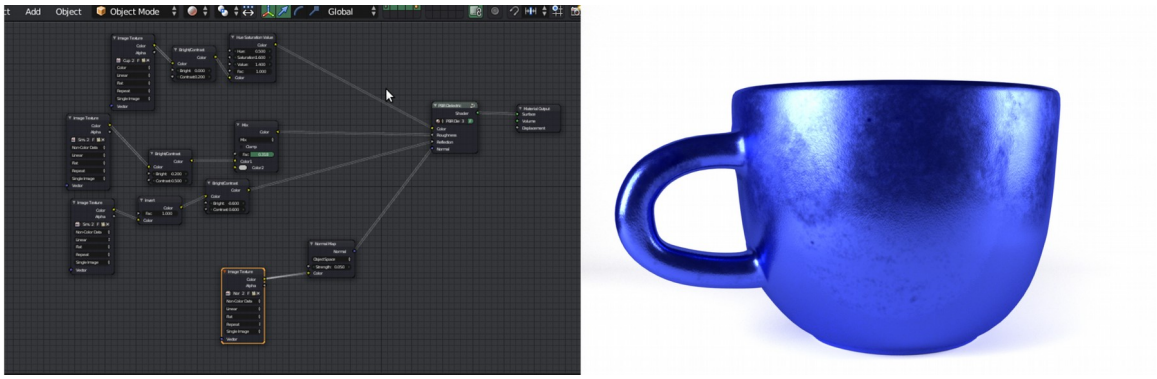


Illustration 12: Final material node setup. Added Diffuse Map and additional smudge with another Gloss Map. Both made in GIMP.

Illustration 13: Final render. Combined cup model with a slightly bigger cup model using the default Glass shader.

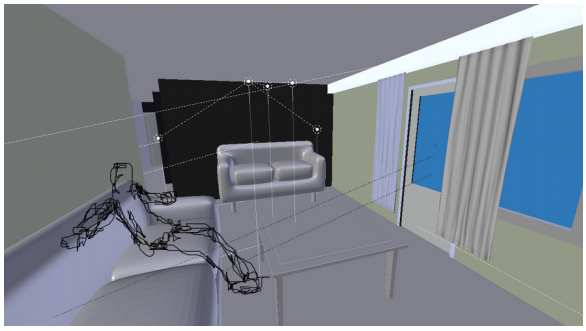
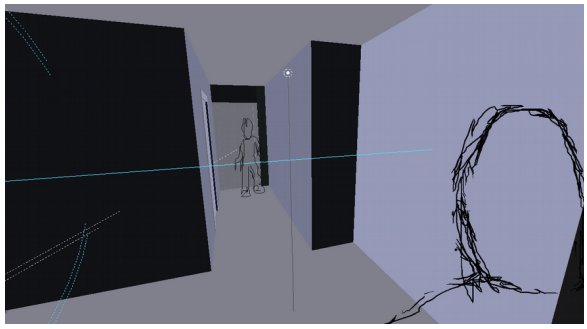
5.4.3 Storyboards and Animatics

For the creation of storyboards or animatics the Open Creative Suite offers a variety of tools. Krita as a drawing application can be used for traditional storyboarding. GIMP can also be used. Especially if a user is more comfortable using photographs than drawing. For this work-flow example I will be using BlenderSB, because the Krita and GIMP methods are fairly basic and similar to using other drawing and image manipulation software.

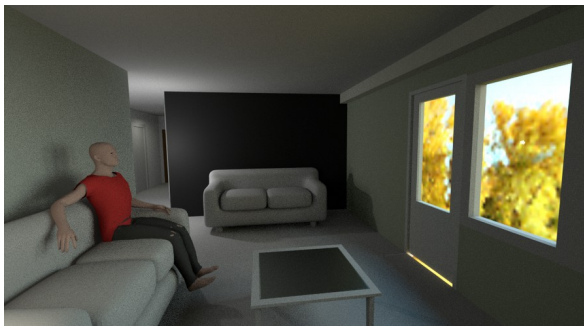
To start on the storyboards there are two ways. First a user can launch BlenderSB and either build a basic model or import a previously created one. To import a blend file to a preexisting layout a user needs to use the append feature. This feature allows a user to import parts of a blend file to another blend file. The second way is to either launch Blender or a preexisting blend file and switch to the storyboard workspace. In this example I will use the first method.

I start by launching BlenderSB and using the append feature to import everything in the objects folder from the blend file of the earlier apartment model. This imports all the objects including their materials. I then save the blend file with the *Save as* feature. This is

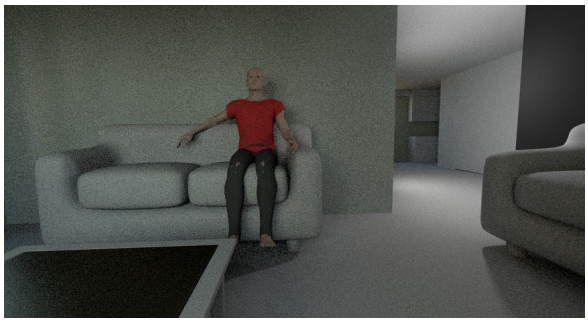
done because BlenderSB is based on a protected blend file that will not allow saving, to prevent overwriting of this file. This only has to be done the first time saving a new file in BlenderSB, after that saving is again possible with the save feature. When the model is imported and saved I can now start working on the storyboard.

To start I use the top view panel to position the camera. After this I use the grease pencil to to sketch in the character, in the main camera panel. When this is done I use the OpenGL render image feature to capture the image and save it. This feature captures the view port instead of the final render image. When this is done, I can move on to the next image. This is a very fast way of creating a simple storyboard.

If a storyboard needs to be more realistic or a user is uncomfortable with drawing, there is the option of using rigged models. To start I again position the camera. After this, I use the armature to pose the model into the correct pose. When this is done I add lights and render the image using Cycles and save it. Then I can move on to the next image. This method is a bit slower than the grease pencil one, but it provides an element of realism. With this method a user can use different camera lenses; choose from a number of presets, what camera to emulate; or test real lighting setups.


To turn a storyboard into an animatic or previsual a user can use the same work-flow as in the previous paragraph, but with animations. These animatics can be very simple short clips with just an animated camera or very complex animations with moving characters, sound, visual effects and so on. This can be very helpful, because a user can, for example. make very realistic tracking, zooming or panning shots; test and plan difficult choreography; or plan shots with a lot of CGI and visual effects. To make this process faster I use Free and Creative Commons licensed motion capture footage that can be found on the web, similar to my furniture database.

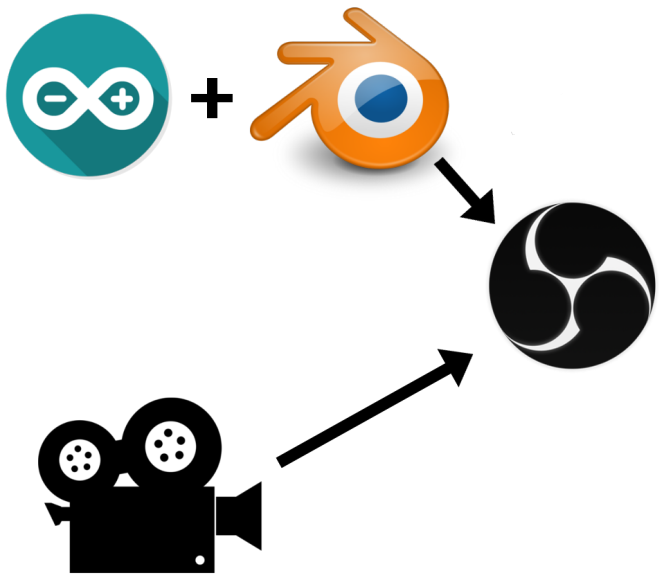
This speeds up the animation of the characters, because basic movements do not need to be animated, for example walking. I would also recommend keeping the picture quality and resolution to a minimum when working on a scene. This is because rendering can be very resource and time-consuming, especially when rendering realistic scenes. To speed up the work I often use the OpenGL render feature when doing test renders of the animation and lower quality cycles or view port rendering.

6 Portable virtual studio

The Portable virtual studio is a project where I attempt to film live video footage with a 3D virtual background and real world foreground, with real time compositing. This is based on the virtual studio system in Aalto University. The point is to use Open Source software and code with a camera and Arduino connected to an IMU sensor to accomplish this. The basic plan is to produce a proof of concept and base to build upon for future development. In this project there are several problems that needed to be solved to produce the proof of concept video. These problems were: what software should the pipeline be built of; how to capture the camera movement; and live composition. A system like this could, for example, be used to previsualize effects on a location shoot involving green screens.

6.1 Building the pipeline

To start the project I first had to solve the pipeline issue. I started with researching the software I could use for the pipeline. To be able to film inside a 3D virtual environment I needed an Open Source application capable of 3D modeling and animation. For this task I chose Blender, because of the variety of features it offers. The problem with Blender is that it does not support live video input to its compositor. For that task I found a program called Open Broadcaster Studio (OBS)[107].



The other issue that needed solving, was how to capture the camera movement. There are many ways motion capture can be done nowadays. There is the use of infrared cameras, IMU sensors and even the Microsoft Kinect[32] can be used for this purpose. While searching for solutions, I came across a tutorial on building a IMU sensor based motion capture suit. This setup was using an Arduino with an IMU sensor to capture movement. Since Arduino is Open Source, this suited my needs perfectly. The final setup is this: the Arduino captures the motion data of the camera and sends it to Blender. There the data is

32 Kinect is a line of motion sensing input devices by Microsoft for Xbox 360 and Xbox One video game consoles and Microsoft Windows PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands.

used to control a virtual camera. Then OBS screen captures the 3D view from Blender and combines it with the footage from the camera.

To test the pipeline I started with trying it out without the moving camera. This test included bringing in two video feeds to OBS. One from the camera and the other one from Blender. Then combining them into a single feed. This proved to be fairly simple and straightforward. To make the background in Blender transparent, I simulated a green screen in the virtual environment and keyed it out in OBS. I made the character's polygon count low by instead of subdividing the mesh adding a subdivision modifier to it, but not applying it. This enabled me to keep the polygon count low but having the mesh appear as high poly. I also baked the texture on the mesh to speed up the render view refresh rate.

The second test consisted of visualizing the movement recorded by the sensor. This I could achieve by using the existing libraries and tutorials provided by Arduino. I loaded the code provided by the libraries to an Arduino using the Arduino IDE[108] and followed a tutorial on their website.

The biggest question I had wasif I could import the data from the sensor to Blender. For this the Open Source and Free software philosophy provided a solution. Some people have already tried similar projects in Blender and released the code under a Free license. Since I myself am new to Python[33] scripting, I would first had to learn how to use this language in Blender, which would have taken a lot of time and effort. But because of other projects releasing their code as Free software, I was able to find a ready-made solution to study and use.

The project I ended up using as a base was a motion capture suit application made by Alvaro Ferran. His project used an Arduino with multiple internal motion sensors to capture the motion of an actor with Blender. The setup uses Pyserial[34] and a Python script to map the sensor data to a virtual character or Armature in Blender. It also uses Blenders game engine to enable real time control of the armature. For the portable virtual studio I was able to edit his work to use a single sensor and move a simple cube in Blender. This then could be used with the sensors attached to a camera and have them perform similar movements in Blender.

33 The primary programming language used by Blender for animation and game logic.

34 Pyserial is a library which provides support for serial connections ("RS-232") over a variety of different devices: old-style serial ports, Bluetooth dongles, infra-red ports, and so on. It also supports remote serial ports via RFC 2217 (since V2.5).

Here is the code I ended up using in Blender:

```
import bge
import math
from math import *
import mathutils
import time
import sys
#sys.path.append("/usr/lib/python3/dist-packages")
import serial
import glob
port=''.join(glob.glob("/dev/ttyACM*"))
ser = serial.Serial(port,115200)
print("connected to: " + ser.portstr)
scene = bge.logic.getCurrentScene()
source = scene.objects
main_arm = source.get('Armature')
ob = bge.logic.getCurrentController().owner
def updateAngles():
        ser.write("a".encode('UTF-8'))
        s=ser.readline()[:-3].decode('UTF-8') #delete ";\r\n"
        angles=[x.split(',') for x in s.split(';')]
        for i in range(len(angles)):
                angles[i] = [float(x) for x in angles[i]]

        trunk = mathutils.Quaternion((angles[0][0],angles[0][1],angles[0][2],angles[0][3]))
        correction = mathutils.Quaternion((0.0, 0.0, 1.0), math.radians(90.0))
        trunk_out = correction*trunk

            ob.channels['trunk'].rotation_quaternion  =  mathutils.Vector([angles[0][0],angles[0][1],angles[0]
[2],angles[0][3]])
        ob.channels['trunk'].rotation_quaternion = trunk_out
        ob.update()
        time.sleep(0.001)
```

To utilize this code I had to do a lot of testing and I did have many errors occur during this. Since I am not an experienced Python programmer, I also had trouble with how to deal with these errors. Here again the Open Source and Free philosophy provided a solution in the form of community. For many problems I was able to find a solution by asking more experienced programmers on the Internet, through the use of online communities like Stack Exchange[35] when I ran into trouble. The Arduino community and documentation also

35 Stack Exchange is a network of question-and-answer websites on topics in varied fields, each site covering a specific topic, where questions, answers, and users are subject to a reputation award process. The sites are modeled after Stack Overflow, a Q&A site for computer programming questions that was

helped out very much, because of their Open Source hardware and software philosophy. By utilizing these communities, reading the open documentation of Blender and Arduino, and reading code from other projects I was able to archive a successful test on the whole pipeline in a fairly short time span. This would have taken considerably longer had I not had access to these resources.

6.2 Building the Camera rig

One of the requirements for this project was that it had to be portable and fairly low budget. This meant that the actual device that captures the movement had to be small, light and cheap. For this there are many solutions in the form of motion sensors and other devices. I ended up using an MPU6050 with an Arduino because these two were easily acquirable. Due to the popularity of Virtual and Augmented Reality in the recent years, there are many other solutions available nowadays. I could just as well have used a Raspberry Pi[36] instead of an Arduino for example. The sensor could also have been replaced with another brand or a more advanced one, but for this test the MPU6050 was adequate.

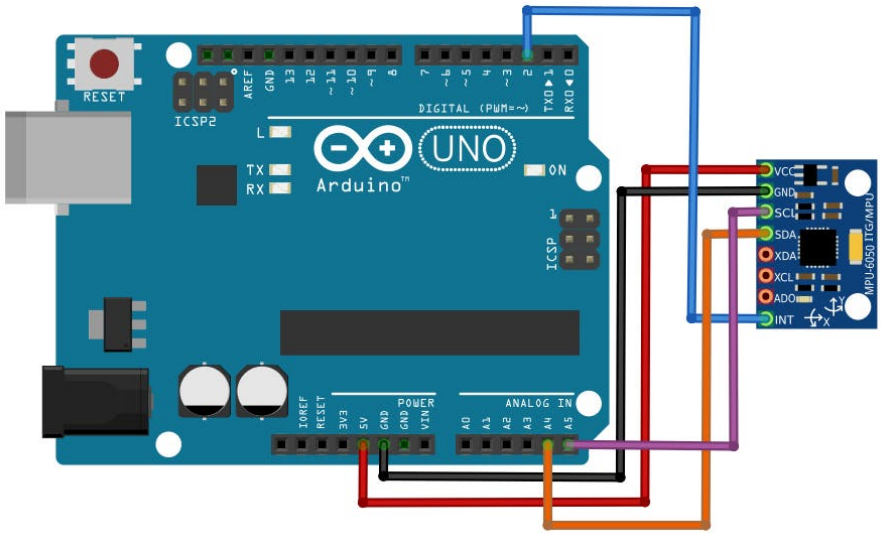


Illustration 14: The MPU6050 wiring.

The reason I decided to use an internal method of motion capture was because external motion capture cameras are usually more expensive and require that the tracking point is visible to at least two infrared cameras to achieve a successful capture. By using an internal sensor the device can be attached to a camera and taken virtually anywhere. In my test the Arduino is attached to a laptop running a GNU/Linux operating system by USB but it could also be used wirelessly by adding Bluetooth or Wi-Fi to the device.

---

the original site in this network.

36 The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries.

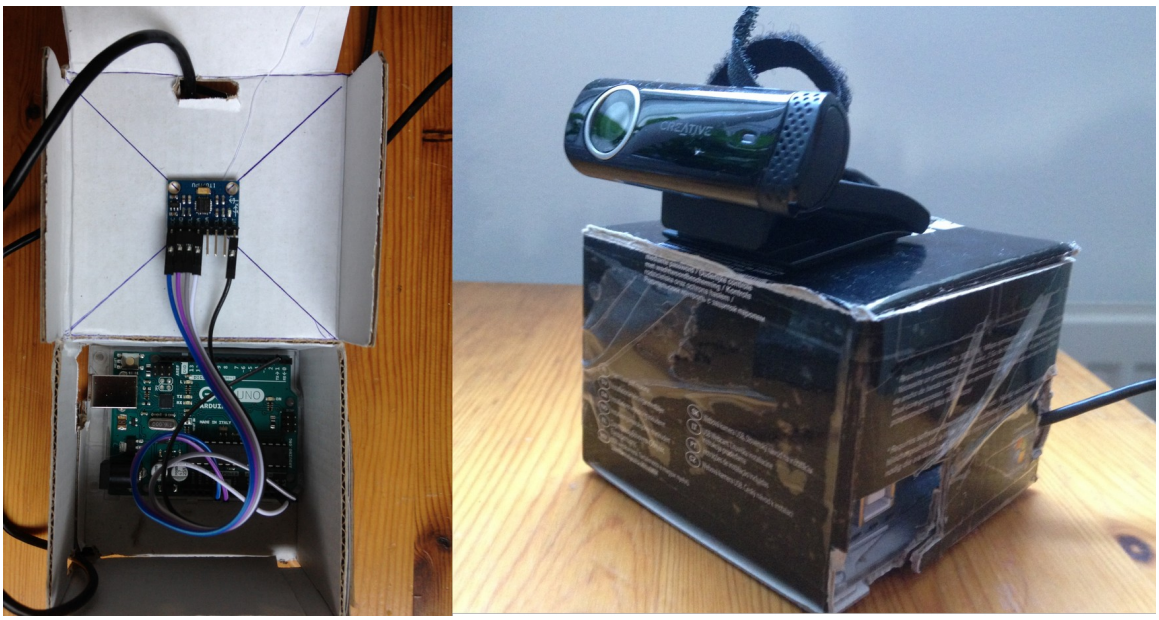


Illustration 15: The prototype and its wiring

The plans for attaching the MPU6050 to the Arduino could be found on Arduino's website and they also provide a tutorial on how to calibrate and get started with it. By using these guides I was able to build a prototype of the sensor and make the reading fairly accurate. I then attached it to a small web camera by creative and had a small camera rig to use in testing the pipeline. For the prototype I used an Arduino UNO but for a finalized product I would probably use an Arduino mini or micro to make the device as small as possible.

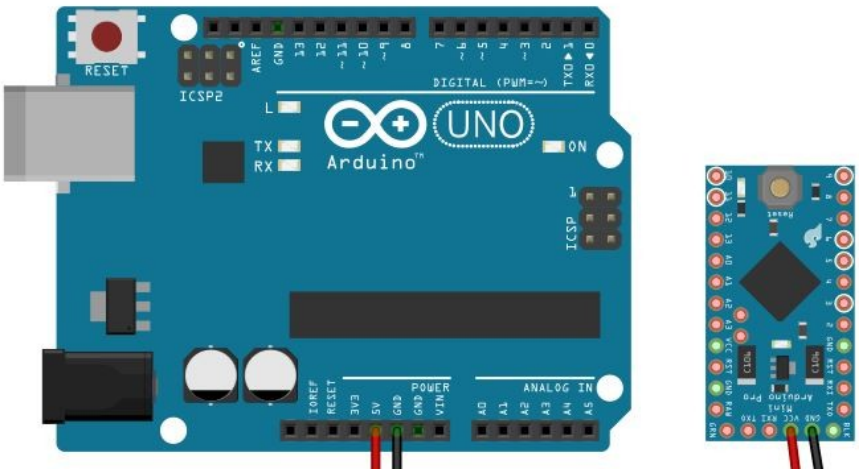


Illustration 16: An Arduino UNO and an Arduino Pro Mini.

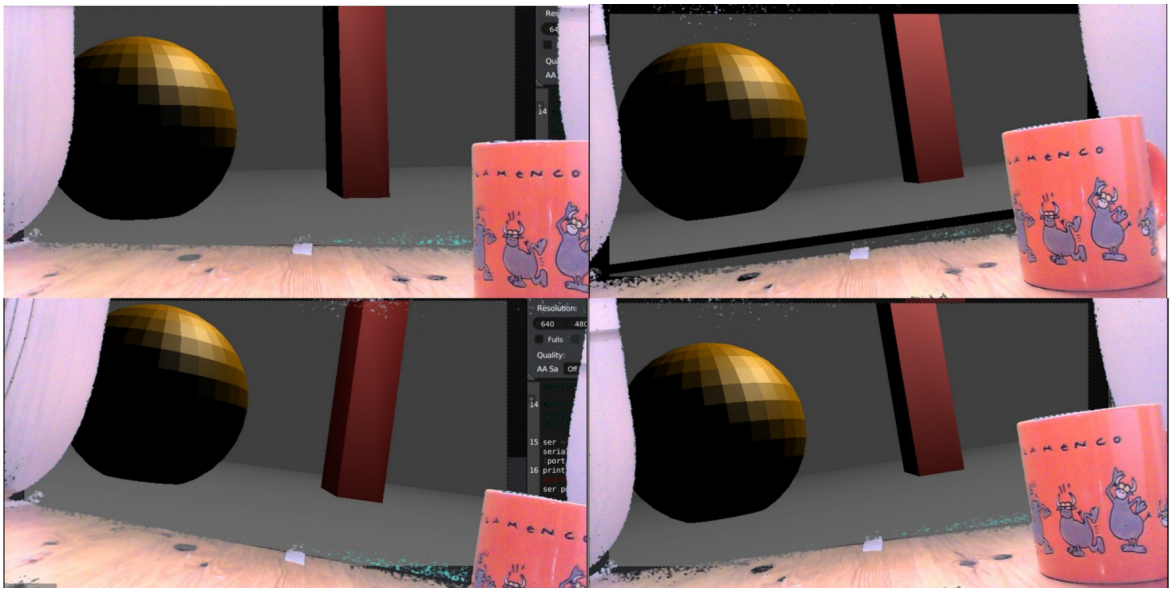
6.3 The proof of concept

After I had the camera device ready, I started working on a proof of concept. This would consist of having a successful test with using the sensor to move a virtual camera in Blender and combining the web-cam footage with Blenders footage in OBS. Before this I

had successfully tested combining still camera footage with Blender and moving a cube with the sensor.

The initial tests proved unsuccessful, because for some reason even if Blender could move an object, moving a virtual camera was not as easy with the sensor. This I was able to work around by instead of moving the camera, I am rotating the world around it. My next issue was accuracy. This had to do with the way an MPU6050 works. Before using this sensor, every unit needs to be calibrated. Luckily many of Arduino's community members have made scripts to help with this and by running one I was able to increase the accuracy by a great deal. To further calibrate and fine tune the movement Alvaro Ferran's Python script also included a correction function. By adjusting these numbers I was able to make the movement very similar to the cameras.

After these issues, I was able to achieve a successful proof of concept. This does not mean that the movement is perfect and I am still experiencing some minor performance issues, but in this successful test I am able to record live camera footage with virtual elements moving in real time with the camera's movement.



6.4 Future plans and improvements

This project is still incomplete and there are many things that need to be solved before a final product can be made. First of all there is the spacial movement of the virtual camera. In my proof of concept I only have a rotating camera but it does not move side to side. For this there are some solutions. First, there is using infrared cameras with the internal sensor to capture the movement. This would increase accuracy but it would affect the portability of the device. Secondly, there is the option of testing if GPS can be used to track the movement. There are also similar devices like the Dead Recon sensor[37] that are used to track spacial movement. Thirdly, there are other sensors that can be used for motion capture that could improve accuracy. For example, flex sensors[38] and magnetometers[39].

The second issue that needs improvement is performance. This could be as simple as using a more powerful computer with a faster USB port or combining OBS functionality with Blender to make the process more streamlined and lighter on the CPU.

Other tests also include trying to use a Raspberry Pi instead of an Arduino to test if the solutions used for this platform are more developed.

As a conclusion, this test was successful in trying the Open Source and Free software philosophy, because my proof of concept would not have been successful without all the community support and previous projects released as Free software.

37 In navigation, dead reckoning, dead-reckoning, ded reckoning (for deduced reckoning), or DR is the process of calculating one's current position by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time and course.

38 A flex sensor or bend sensor is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer.

39 A magnetometer is an instrument that measures magnetism—either magnetization of magnetic material like a ferromagnet, or the direction, strength, or the relative change of a magnetic field at a particular location.

7 Conclusion

The open nature of sharing content, techniques and tools, is something I as a designer find very important. These philosophies and ideas are in my opinion, essential for the improvement and development of the film and media field. For example, the technical aspects of this field is something that has been growing consistently for many years. In this thesis I wanted to explore these philosophies and alternatives, in hope of providing alternatives to the way many of these aspects are achieved today.

I asked in the introduction of this thesis if “Open Source and Free software can be used for professional film production, media production and education?” For me this has been proven. As a viable alternative for many industry standards, Open Source and Free software could provide an option. In my opinion, the two projects I have described in this thesis support this. In my work as a production designer I can do all my design work on a completely Open Source and Free pipeline, like the Open Creative Suite. While working on the Portable virtual studio, the Open Source and Free philosophy helped me immensely to achieve a proof of concept. Some other projects also encourage this, like Tangent Animations feature film *Ozzy (2016).*

These are of course very small projects and testing the Open Source and Free philosophy on a grander scale could prove interesting. For example, in a production company. In a small country like Finland we as film makers could benefit from a more collaborative way of working.

I believe that Open Source, Free and Creative Commons do have a place in the professional field and that we as film-makers should support these things. Not only for the benefits the software offers, but as other creative content platforms become more popular. Streaming platforms like YouTube and Netflix have for instance become very popular and YouTube is for example filled with Creative Commons licensed content.

For my peers and colleagues I hope this thesis has provided them with an alternative work-flow more suitable for freelancers, students and smaller production companies.

"I could have made money this way, and perhaps amused myself writing code. But I knew that at the end of my career, I would look back on years of building walls to divide people, and feel I had spent my life making the world a worse place." - Richard M. Stallman

Sources

[1]: Microsoft Windows 1.0, 1985, Operating System, Windows
[2]: Richard Stallman/Linus Torvalds/Community, The Gnu/Linux Operating System, 1992, Operating System, GNU/Linux
[3]: Canonical Ltd., Ubuntu community, Ubuntu Desktop, 2004, Operating System, GNU/Linux
[4]: n/a, Microsoft Windows Vista, 2007, Operating system, Windows
[5]: n/a, Microsoft Windows 7, 2009, Operating System, Windows
[6]: , Adobe Photoshop, 1990, Image editor, Windows/Mac OS
[7]: Brett Smith, A Quick Guide to GPLv3, Retrieved 18.12.2016
[8]: Licenses & Standards, Retrieved 18.12.2016, https://opensource.org/licenses
[9]: About Free Software Foundation, Retrieved 18.12.2016, http://www.fsf.org/about/
[10]: About Open Source Initiative, Retrieved 18.12.2016, https://opensource.org/about
[11]: Jupiter Broadcasting, Retrieved 18.12.2016, http://www.jupiterbroadcasting.com/support-us/
[12]: Blender Projects, Retrieved 18.12.2016, https://www.blender.org/features/projects/
[13]: OpenEXR, Retrieved 18.12.2016, http://www.openexr.com/
[14]: Brainstorm eStudio, n/y, real time 3D graphics and virtual studio engine, Windows
[15]: J. A. Okun & S. Zwerman, The VES Handbook of Visual Effects: Industry Standard VFX Practises and Procedures, 2010
[16]: Jim Fruchterman, Is your open source security software less secure?, Retrieved 18.12.2016 https://opensource.com/business/15/5/why-open-source-means-stronger-security
[17]: Open Source for Business, Retrieved 18.12.2016, https://opensource.org/osforbusiness
[18]: Linus Torvalds/community, The Linux Kernel 0.01, 1991, Operating System Kernel, Unix
[19]: Debian, 1993, Operating System, GNU/Linux
[20]: The Ubuntu story, Retrieved 18.12.2016, https://www.ubuntu.com/about/about-ubuntu
[21]: Free Software and Education, Retrieved 18.12.2016, https://www.gnu.org/education/education.html
[22]: Final Cut Pro, 1998, Video editing software, Mac OS
[23]: Adobe Premiere Pro, 1991, Video editor, Windows/Mac OS
[24]: Avid Media Composer, 1992, Video editor, Windows/Mac OS
[25]: Alias Systems Corporation, Autodesk Maya, 1998, 3D computer graphics, Windows/Mac OS/RHEL/CentOS/Fedora
[26]: Adobe Illustrator, 1987, Vector graphics editor, Windows/Mac OS
[27]: Ton Roosendaal, Blender, 1995, 3D computer graphics software, Windows/Mac OS/Linux
[28]: J. Bell, K. Zorniak, Tangent Animation: Making Ozzy with Blender, 2016 https://www.blender.org/conference/2016/presentations/311, https://www.youtube.com/watch?v=CqSEtTF8hPE
[29]: 2017, Retrieved 27.04.2017, http://www.tuotantokannustin.fi/
[30]: Final Cut Pro 7, 2009, Video editor, Mac OS
[31]: Final Cut Pro X, 2011, Video editor, Mac OS
[32]: Chris Foresman, More Final Cut Pro X fallout: top reality producer ditches Apple for Avid, 2012, Retrieved 10.01.2017, http://arstechnica.com/apple/2012/01/more-fcpx-fallout-top-reality-producer-ditches-apple-for-avid/
[33]: Unity Desktop, 2010, Graphical shell, Linux
[34]: GNOME Desktop, 1999, Desktop environment, Linux/Unix
[35]: Ubuntu GNOME, 2012, Operating System, Linux
[36]: Android, 2008, Mobile operating system, 32- and 64-bit ARM/x86/x86-64/MIPS/MIPS64
[37]: Mac OS, 1984, Operating system, Unix
[38]: Red Hat Linux, 1995, Operating system, Linux
[39]: A. Khamooshi , Breaking Down Apple’s iPhone Fight With the U.S. Government, 2016, Retrieved 23.04.2017, https://www.nytimes.com/interactive/2016/03/03/technology/apple-iphone-fbi-fight-explained.html?_r=0
[40]: Microsoft Privacy Statement, Retrieved 23.04.2017, https://privacy.microsoft.com/en-US/privacystatement

[41]: FreeBSD, 1993, Operating system, Unix
[42]: Judd Vinet/Aaron Griffin/Community, Arch Linux, 2002, Operating system, Linux
[43]: Community Enterprise Operating System, 2004, Operating System, Linux
[44]: Ikey Doherty, Solus Project, 2012, Operating system, Linux
[45]: Jürgen Riegel/Werner Mayer/Yorik van Havre, FreeCad, 2002, Computer aided design, Windows/Mac OS/Linux/Unix
[46]: AutoCAD, 1982, Computer aided design, Windows/Mac OS/IOS/Android
[47]: Hendrik vermooten/Hein Oosthuizen, TurboCAD, n/y, Computer aided design, Windows/Mac OS
[48]: Last Software/Google, ScechUp Pro, 2000, 3d computer graphics, Windows/Mac OS
[49]: Alex Fitzpatrick, A judge ordered microsoft to split. Heres why it's still a single company, Retrieved 11.01.2017, http://time.com/3553242/microsoft-monopoly/
[50]: Softimage Co., Softimage, 2000, 3D computer Graphics, Windows/Linux
[51]: Softimage final release announcement, Retrieved 11.01.2017, http://www.autodesk.com/products/softimage/overview
[52]: 3D Studio Max, 1990, 3D computer graphics, Windows
[53]: Altsys Corporation, FreeHand, 1988, Vector graphics editor, Windows/Mac OS
[54]: Ubuntu Touch, 2011, Mobile operating system, ARM/x86
[55]: Snappy package manager, 2014, Package manager, Linux
[56]: About the department, 2014, Retrieved 23.04.2017, http://elo.aalto.fi/en/about/
[57]: HANDS-ON FILM & ACTING SCHOOL, 2017, Retrieved 28.04.2017, https://www.nyfa.edu/
[58]: Learn the Art of 3D Animation, 2017, Retrieved 28.04.2017, https://www.nyfa.edu/animation-school/programs/
[59]: 3D Animation + Visual Effects curriculum, 2017, Retrieved 28.04.2017, https://vfs.edu/programs/3D-animation-vfx/curriculum
[60]: Production Design Program, 2017, Retrieved 28.04.2017, https://nfts.co.uk/our-courses/masters/production-design
[61]: Spencer Kimball/Peter Mattis, Gnu Image Manipulation Program, 1995, Image editor, Windows/Mac OS/BSD/Solaris/Linux/AmigaOS 4
[62]: Proprietary Software Is Often Malware, n/y, Retrieved 01.05.2017, https://www.gnu.org/philosophy/proprietary.html
[63]: Richard Stallman/Community, The GNU Operating System, , Operating system, IA-32 (with Hurd kernel only) and Alpha/ARC/ARM/AVR32/Blackfin/C6x/ETRAX CRIS/Hexagon/Itanium/ M32R/m68k/META/Microblaze/MIPS/MN103/OpenRISC/PA-RISC/PowerPC/s390/S+core/ SuperH/ SPARC/TILE64/Unicore32/x86/Xtensa (with Linux-libre kernel only)
[64]: Richard Stallman, GNU Emacs, 1985, Text editor, GNU/Linux
[65]: S. Williams, R.M. Stallman, Freedom 2.0: Richard Stallman and the Free Software Revolution, 2010
[66]: Free System Distribution Guidelines, Retrieved 24.02.2017, https://www.gnu.org/distros/free-system-distribution-guidelines.html
[67]: B. M. Hill, The computer in my pocket, 2010, Retrieved 24.02.2017, https://www.fsf.org/working-together/next-steps/computer-in-my-pocket
[68]: J. Gay, Respects Your Freedom hardware product certification, Retrieved 24.02.2017, https://www.fsf.org/resources/hw/endorsement/respects-your-freedom
[69]: N. Newman, The importance of trust in news provision, 2016, Retrieved 23.04.2017, http://digitalnewsreport.org/survey/2016/overview-key-findings-2016/
[70]: J. Kauppinen, Pääministeri Sipilä vaiensi Ylen: Uutisten johto hyllytti Sipilä-jutut – Ruben Stiller sai varoituksen, 2016, Retrieved 23.04.2017, https://suomenkuvalehti.fi/jutut/kotimaa/paaministeri-sipila-vaiensi-ylen-uutisten-johto-hyllytti-sipila-jutut-ruben-stiller-sai-varoituksen/
[71]: Meridith Levinson, Industrial Light and Magic Replacing Unix With Linux, 2001, Retrieved 30.04.2017, http://www.cio.com/article/2441140/linux/industrial-light-and-magic-replacing-unix-with-linux.html
[72]: Copyright act, (607/2015) Section 43, page 27
[73]: 2017, Retrieved 26.02.2017, https://distrowatch.com/search.php?status=All
[74]: The Brisk Menu, 2016, Desktop menu, GNU/Linux

[75]: The Mate Desktop, 2011, Desktop Environment, GNU/Linux
[76]: Brisk menu, 2016, Retrieved 26.02.2017, https://solus-project.com/2016/12/11/this-week-in-solus-install-40/
[77]: Ubuntu MATE, 2014, Operating system, GNU/Linux
[78]: Universal Scene Description, n/y, Pipeline tool,
[79]: Krita, 2005, Raster graphics editor, Windows/Mac OS/Linux
[80]: Digital Domain, NUKE, 1993, Compositing application , Windows/Mac OS/Linux
[81]: About krita, History, Retrieved 23.04.2017, https://krita.org/en/about/history/
[82]: Celtx, n/y, Pre-production software, Cloud based
[83]: Osku Salmela/Anil Gulecha, Trelby, 2003, Screenwriting Software, Windows/Linux
[84]: Apache Open Office, 2012, Office Program, Windows/Mac OS/Linux
[85]: StarDivision, LibreOffice, 2011, Office program, Windows/Mac OS/ Linux/ FreeBSD/ NetBSD/ Android(Viewer only)
[86]: Microsoft Office, 1990, Office program, Windows/Mac OS
[87]: Telegram, 2013, Instant messaging service, Windows Phone/Android/Mac OS/IOS/WebApp/Linux
[88]: Thorvald Natvig, Mumble, 2005, Voice over IP application, Windows/Mac OS/Linux/Android/IOS
[89]: Priit Kasesalu/Jaan Tallinn, Skype, 2003, Instant messaging application, Windows/MacOS/Linux/Blackberry OS/IOS/WatchOS/Windows Phone/HoloLens/Xbox One
[90]: Nextcloud, 2016, Client-server software, Server: Linux, Clients: Windows/Mac OS/Linux/Android/IOS
[91]: OwnCloud, 2011, Client-server Software, Server: Linux, Clients: Windows/Mac OS/Linux/Android/IOS
[92]: Dropbox, 2007, File hosting service, Windows/Mac OS/Linux/Windows Phone/IOS/Android
[93]: Google Drive, 2012, File hosting service, Web Service
[94]: Florian Höch, DisplayCAL, n/y, Display calibration software, Windows/Mac OS/Linux
[95]: GOME Color Manager, n/y, Color management tool, Linux
[96]: Johannes Hanika, Darktable, 2009, Image post-production, Mac OS/Linux/FreeBSD/Solaris/Windows(unofficial)
[97]: Jason Wood, Kdenlive, 2002, Video editor, Windows/FreeBSD/Linux
[98]: Dominic Mazzoni/Roger Dannenberg, Audacity, 2000, Audio editor/recorder, Windows/Mac OS/Linux/Unix
[99]: Paul Davis, Ardour, 2005, Audio editor/recorder, Windows/Mac OS/Linux/FreeBSD
[100]: Rosegarden, 1993, Digital audio workstation program, BSD/Linux
[101]: Paul Giblock/Tobias Doerffel, Linux MultiMedia Studio, 2004, Digital audio workstation software, Windows/GNU/Linux/Mac OS
[102]: Rui Nuno Capela, Qtractor, 2005, Digital audio workstation software, Linux
[103]: Lightworks, 1989, Video editor, Windows/Mac OS/Linux
[104]: Ubuntu Studio, 2007, Operating system, Linux
[105]: Elementary OS, 2011, Operating system, Linux
[106]: Olivier Fourdan, Xfce, 1996, Desktop environment, Linux
[107]: Hugh "Jim" Bailey, Open Broadcaster Software, 2012, Streaming/recording Software, Windows/Mac OS/Linux
[108]: Arduino IDE, 2003, Integrated development environment, Windows/Mac OS/Linux

Films

Harry Potter 2001-2011, Screenplay: Steve Kloves (1-4,6-8), Michael Goldenberg (5). Direction: Chris Columbus (1-2), Alfonso Cuaròn (3), Mike Newell (4), David Yates (5-8). Produced: David Heyman, Chris Columbus (3), Mark Radcliffe (3), David Baron (5-8), J.K. Rowling (7-8). Heyday Films, 1492 Pictures (1-3), Patalex IV Productions (4), Warner Bros. Pictures. 1179 min.

Men in Black II 2002, Screenplay: Robert Gordon, Barry Fanaro. Direction: Barry Sonnenfeld. Produced: Walter F. Parkes, Laurie MacDonald. Amblin Entertainment, Parkes/MacDonald Production, Columbia Pictures. 89 min.

Gangs of New York 2002, Screenplay: Jay Cocks, Steven Zaillian, Kenneth Lonergan. Direction: Martin Scorsese. Produced: Alberto Grimaldi, Harvey Weinstein. Alberto/Grimaldi Productions, Initial Entertainment Group, Miramax. 168 min

Signs 2002, Screenplay: M. Night Shyamalan. Direction: M. Night Shyamalan. Produced: M. Night Shyamalan, Frank Marshall, Kathleen Kennedy, Sam Mercer. Touchstone Pictures, Blinding Edge Pictures, The Kennedy/Marshall Company, Buena Vista Pictures. 107 min.

Ozzy 2016, Screenplay: Juan Ramòn Ruiz de Somavia. Direction: Alberto Rodriguez, Nacho La Casa. Produced: Jeff Bell, Ibon Cormenzana, Phyllis Laing, Ken Zorniak. Arcadia Motion Pictures, BD Animation, Capitán Araña, Pachacamac Films, Tangent Animation. 90 min.

Star Wars 1977, Screenplay: George Lucas. Director: George Lucas. Production: Gary Kurtz. Lucasfilm Ltd. 20th Century Fox. 121 min.

Jurassic Park 1993, Screenplay: Michael Crichton, David Koepp. Direction: Steven Spielberg. Production: Kathleen Kennedy, Gerald R. Molen. Amblin Entertainment, Universal Pictures. 127 min.

Tears of Steel 2012, Screenplay: Ian Hubert. Direction: Ian Hubert. Produced: Ton Roosendaal. Blender Foundation. 12 min.

The Secret Life of Pets 2016, Screenplay: Brian Lynch, Cinco Paul, Ken Daurio. Direction: Chris Renaud. Produced: Chris Meledandri, Janet Healy. Illumination Entertainment, Universal Pictures. 87 min.

The Wolf of Wall Street 2013, Screenplay: Terence Winter. Direction: Martin Scorsese. Produced: Martin Scorsese, Leonardo DiCaprio, Riza Aziz, Joey McFarland, Emma Tillinger Koskoff. Red Granite Pictures, Appian Way Productions, Sikelia Productions, EMJAG Productions, Paramount Pictures. 179 min

Pulp Fiction 1994, Screenplay: Quentin Tarantino. Direction:Quentin Tarantino. Produced: Lawrence Bender. A Band Apart, Jersey Films, Miramax Films. 154 min.

The Kings Speech 2010, Screenplay: David Seidler. Direction: Tom Hopper. Produced: Iain Canning, Emile Sherman, Gareth Unwin. UK Film Council, See-Saw Films, Bedlam Productions, The Weinstein Company, Momentum Pictures. 119 min.

Appendix

List of Open Source and Free Media production software				
Name	Type of software	License	Price	Supported platforms
Libre Office	Office	GPL v3	Free	Windows, Mac OS, GNU/Linux
Apache Open Office	Office	Apache License 2.0	Free	Windows, Mac OS, GNU/Linux
Neo Office	Office	GPL	15€	Mac OS
Scribus	Publishing/Page layout	GPL	Free	Windows, Mac OS, GNU/Linux
Trelby	Screen writing	GPL	Free	Windows, Linux
Celtx Desktop	Screen writing	Motzilla Public License v2.0	Free	Windows, Mac OS, GNU/Linux
GIMP	Image editing	GPL v3	Free	Windows, Mac OS, GNU/Linux
Inkscape	Drawing	GPL v3	Free	Windows, Mac OS, GNU/Linux
Krita	Drawing	GPL v3	Free	Windows, Mac OS, GNU/Linux
Darktable	Image editing	GPL v3	Free	Windows, Mac OS, GNU/Linux
Hugin	Panorama creator	GPL v2	Free	Windows, Mac OS, GNU/Linux
Blender	3D creation suite	GPL v2	Free	Windows, Mac OS, GNU/Linux
Makehuman	Character modelling app	AGPL	Free	Windows, Mac OS, GNU/Linux
Audacity	Audio recording/Editing	GPL v2	Free	Windows, Mac OS, GNU/Linux
Ardour	Audio recording/Editin	GPL v2	Free	Windows, Mac OS, GNU/Linux

	g			
Rosegarden	Music composition	GPL	Free	Linux
MuseScore	Music composition	GPL	Free	Windows, Mac OS, GNU/Linux
Natron	Compositing	GPL v2	Free	Windows, Mac OS, GNU/Linux
KdenLive	Video Editing	GPL v2	Free	Windows, Linux
Flowblade	Video Editing	GPL v3	Free	Linux
Pitivi	Video Editing	LGPL	Free	Linux
Shotcut	Video Editing	GPL v3	Free	Windows, Mac OS, GNU/Linux
OpenShot	Video Editing	GPL v3	Free	Windows, Mac OS, GNU/Linux
Synfig Studio	2D Animation	GPL	Free	Windows, Mac OS, GNU/Linux
Valentina	Sewing Pattern App	GPL v3	Free	Windows, Mac OS, GNU/Linux
Open Broadcaster Studio	Streaming/Recording App	GPL v2	Free	Windows, Mac OS, GNU/Linux

Links to the projects

The Open Creative Suite:

https://github.com/Bjaza/OC_suite

The Portable virtual studio:

https://github.com/Bjaza/mpu6050Arduino-blender